System Administrator's Reference Manual

Commands(1M)

Introduction

System Administrator's Reference Manual

intro_sarm1(1M)

NAME

`intro_sarm1 -` introduction to commands and application programs

DESCRIPTION

This section describes commands that are mainly used for system maintenance and administration purposes, listing them in alphabetical order.

Since the commands for the architecture of the virtual file system have been restructured, there are several instances of multiple manual pages with the same name. For example, there are two manual pages dealing with the command `mount(1M)` . In these cases, the first of the multiple pages describes the syntax and options of the generic command, i.e. those options applicable to all FSTs (file system types). The functionality of the FST-specific modules for the command is described below, and the name of the FST in question is also displayed. Note that the administrator should not attempt to call these modules directly. The generic command provides a common interface to all of them. Thus the FST-specific manual pages should not be viewed as describing distinct commands, but rather as detailing those aspects of a command that are specific to a particular type of file system.

SYNOPSIS

Options and arguments of the commands are specified according to the following syntax:

`name` [`–option(s)`] [*cmdarg*(*s*)] [`nvarg(s)`]

The different syntax elements have the following meaning:

[□] Selectable option or argument

`name` Name of an executable file (command)

`option`
(always with a preceding "–"): `noargletter(s)` or: `argletter`ERROR*optarg*

`noargletter`
One or more single letters each representing an option *without* an argument (e.g. `–i` or `–dna`).

`argletter`
A single letter representing an option requiring an argument (*optarg*), e.g. `t` in `–tERROR`*file*.

optarg
An argument (character string) satisfying preceding `argletter` (e.g. *file* in `–tERROR`*file*).

cmdarg
Variable argument(s) (character string) for a command, e.g. a pathname, directory, target device etc.; "–" indicates the standard input. Examples:

`devfree` *key* [*device* [...]] or: `colltbl` [*file* | –]

`nvarg`
One or more arguments that cannot be modified, e.g. `on□|□off`, `demand`

... Repetition: One or more options or arguments can be specified several times. Example: [*attribute* ...]

| Alternative: Only one option or one argument of several commands can be specified. Example: `–v` |`–p` |`–a`

{□} Grouping: At least one option or argument from this group must be specified. Example: {*special* | *mount_point*}

DIAGNOSTICS

When terminated, each command returns 0 for normal termination and a value other than zero to indicate problems such as erroneous parameters, bad or inaccessible data, or any other difficulty coping with the task at hand. This value is called "exit code" or "return code", and is described only where special conventions are involved.

NOTE

Unfortunately, not all commands adhere to the standard syntax.

SEE ALSO

`getopts`(1), `exit`(2), `wait`(2), `getopt`(3C).

accept(1M) - autopush(1M)

accept(1M)

NAME

`accept, reject` - accept or reject print requests

SYNOPSIS

`accept` *destinations*

`reject` [`–r` *reason*] *destinations*

DESCRIPTION

`accept` allows the queuing of print requests for the named *destinations*. A *destination* can be either a printer or a class of printers. Run `lpstat –a` to find the status of *destinations*.

`reject` prevents queuing of print requests for the named *destinations*. A *destination* can be either a printer or a class of printers. (Run `lpstat –a` to find the status of *destinations*.) The following option is useful with `reject`.

`–r` *reason*

Assign a *reason* for rejection of requests. This *reason* applies to all *destinations* specified. *reason* is reported by `lpstat –a`. It must be enclosed in quotes if it contains blanks. The default reason is `unknown reason` for existing destinations, and `new destination` for destinations just added to the system but not yet accepting requests.

FILES

`/var/spool/lp/*`

SEE ALSO

`enable`(1), `lp`(1), `lpstat`(1) `lpadmin`(1M), `lpsched`(1M).

System Administrator's Reference Manual

acct(1M)

NAME

acct:
acctdisk, acctdusg, accton, acctwtmpclosewtmp, utmp2wtmp - overview of accounting andmiscellaneous accounting commands

SYNOPSIS

/usr/lib/acct/acctdisk

/usr/lib/acct/acctdusg [–u *file*] [–p *file*]

/usr/lib/acct/accton [*file*]

/usr/lib/acct/acctwtmp "*reason*"

/usr/lib/acct/closewtmp

/usr/lib/acct/utmp2wtmp

DESCRIPTION

Accounting software is structured as a set of tools (consisting of both C programs and shell procedures) that can be used to build accounting systems. acctsh(1M) describes the set of shell procedures built on top of the C programs.

Connect time accounting is handled by various programs that write records into /var/adm/wtmp, as described in utmp(4) . The programs described in acctcon(1M) convert this file into session and charging records, which are then summarized by acctmerg(1M) .

Process accounting is performed by the Reliant UNIX system kernel. Upon termination of a process, one record per process is written to a file (normally /var/adm/pacct). The programs in acctprc(1M) summarize this data for charging purposes; acctcms(1M) is used to summarize command usage. Current process data may be examined using acctcom(1) .

Process accounting and connect time accounting (or any accounting records in binary format) can be merged and summarized into total accounting records by acctmerg. prtacct [see acctsh(1M)] is used to format any or all accounting records.

acctdisk reads lines that contain user ID, login name, and number of disk blocks and converts them to total accounting records that can be merged with other accounting records.

acctdusg reads its standard input (usually from find / –print) and computes disk resource consumption (including indirect blocks) by login. If –u is given, records consisting of those filenames for which acctdusg charges no one are placed in *file* (a potential source for finding users trying to avoid disk charges). If –p is given, *file* is the name of the password file. This option is not needed if the password file is /etc/passwd.

accton alone turns process accounting off. If *file* is given, process accounting is turned on and the kernel process accounting records to *file* [see acct(2) and acct(4)].

acctwtmp writes a utmp(4) record to its standard output. The record contains the current time and a string of characters that describe the *reason*. A record type of ACCOUNTING is assigned [see utmp(4)]. *reason* must be a string of 11 or fewer characters, numbers, $, or spaces.

For example, the following are suggestions for use in reboot and shutdown procedures, respectively:

```
acctwtmp "acctg on" >> /var/adm/wtmp
acctwtmp "acctg off" >> /var/adm/wtmp
```

For each user currently logged on, closewtmp puts a false DEAD_PROCESS record in the /var/adm/wtmp file. runacct [see runacct(1M)] uses this false DEAD_PROCESS record so that the connect accounting procedures can track the time used by users logged on before runacct was invoked.

For each user currently logged on, runacct uses utmp2wtmp to create an entry in the file /var/adm/wtmp, created by runacct. Entries in /var/adm/wtmp enable subsequent invocations of runacct to account for

connect times of users currently logged in.

FILES

`/etc/passwd`
: used for login name to user ID conversions

`/usr/lib/acct`
: holds all accounting commands listed in sub-class 1M

`/var/adm/pacct`
: current process accounting file

`/var/adm/wtmp`
: login/logoff history file

SEE ALSO

`acctcom`(1), `acctcms`(1M), `acctcon`(1M), `acctmerg`(1M), `acctprc`(1M), `acctsh`(1M), `fwtmp`(1M), `runacct`(1M), `acct`(2), `acct`(4), `utmp`(4).

System Administrator's Reference Manual

acctcms(1M)

NAME

`acctcms` - command summary from per-process accounting records

SYNOPSIS

`/usr/lib/acct/acctcms [ –a [–p ] [ –o ]] [ –c ] [ –j ] [ –n ] [ –r` *size* `] [ –s ] [ –t ]` *files*

DESCRIPTION

`acctcms` reads one or more *files*, normally in the form described in `acct(4)` . It adds all records for processes that executed identically-named commands, sorts them, and writes them to standard output, normally using an internal summary format.

OPTIONS

`–a` Print output in ASCII rather than in the internal summary format. The output includes command name, number of times executed, total kcore-minutes, total CPU minutes, total real minutes, mean size (in K), mean CPU minutes per invocation, "hog factor", characters transferred, and blocks read and written, as in `acctcom(1)` . Output is normally sorted by total kcore-minutes.

`–c` Sort by total CPU time, rather than total kcore-minutes.

`–j` Combine all commands invoked only once under "***other".

`–n` Sort by number of command invocations.

`–r` *size*
By default, the hash table for `acctcms` can hold 5001 entries. If this is not large enough, the message: "acctcms: Hash table overflow" appears. You can use the `–r` *size* option to increase the size of this table.

`–s` Any filenames encountered hereafter are already in internal summary format.

`–t` Process all records as total accounting records. The default internal summary format splits each field into prime and non-prime time parts. This option combines the prime and non-prime time parts into a single field that is the total of both, and provides upward compatibility with old (i.e. UNIX System V) style `acctcms` internal summary format records.

The following options may be used only with the `–a` option.

`–p` Output a prime-time-only command summary.

`–o` Output a non-prime (offshift) time only command summary.

When `–p` and `–o` are used together, a combination prime and non-prime time report is produced. All the output summaries will be total usage except number of times executed, CPU minutes, and real minutes, which will be split into prime and non-prime.

A typical sequence for performing daily command accounting and for maintaining a running total is:

```
acctcms file ... > today
cp total previoustotal
acctcms –s today previoustotal > total
acctcms –a –s today
```

NOTES

Unpredictable output results if `–t` is used on new style internal summary format files, or if it is not used with old-style internal summary format files.

SEE ALSO

`acctcom(1)` , `acct(1M)` , `acctcon(1M)` , `acctmerg(1M)` , `acctprc(1M)` , `acctsh(1M)` , `fwtmp(1M)` ,

runacct(1M), acct(2), acct(4), utmp(4).

System Administrator's Reference Manual

acctcon(1M)

NAME

acctcon, acctcon1, acctcon2 - connect-time accounting

SYNOPSIS

/usr/lib/acct/acctcon [*options*]

/usr/lib/acct/acctcon1 [*options*]

/usr/lib/acct/acctcon2

DESCRIPTION

acctcon converts a sequence of login/logoff records to total accounting records. login/logoff records are read from standard input. The file /var/adm/wtmp is usually the source of the login/logoff records, however, because it may contain corrupted records or system date changes, it should first be fixed using wtmpfix. The fixed version of file /var/adm/wtmp can then be redirected to acctcon. The binary records are written to standard output.

OPTIONS

–l *file*

file is created to contain a summary of line usage showing line name, number of minutes used, percentage of total elapsed time used, number of sessions charged, number of logins, and number of logoffs. This file helps track line usage, identify bad lines, and find software and hardware oddities. Hangup, termination of login(1) and termination of the login shell each generate logoff records, so that the number of logoffs is often three to four times the number of sessions [see init(1M) and utmp(4)].

–o *file*

file is filled with an overall record for the accounting period, giving starting time, ending time, number of reboots, and number of date changes.

acctcon is a combination of the programs acctcon1 and acctcon2. acctcon1 converts login/logoff records, taken from the fixed /var/adm/wtmp file, to ASCII output. acctcon2 reads the ASCII records produced by acctcon1 and converts them to binary records. acctcon1 can be used with the –l and –o options, described above, as well as with the following options:

–p Print input only, showing line name, login name, and time (in both numeric and date/time formats).

–t acctcon1 maintains a list of lines where users are logged in. When it reaches the end of its input, it emits a session record for each line that still appears to be active. It normally assumes that its input is a current file, so that it uses the current time as the ending time for each session still in progress. The –t flag causes it to use, instead, the last time found in its input, thus assuring reasonable and repeatable numbers for non-current files.

EXAMPLES

The acctcon command is typically used as follows:

```
acctcon –l lineuse –o reboots < tmpwtmp > ctacct
```

The acctcon1 and acctcon2 commands are typically used as follows:

```
acctcon1 –l lineuse –o reboots < tmpwtmp | sort +1n +2 > ctmp
acctcon2 < ctmp > ctacct
```

NOTES

The line usage report is confused by date changes. Use wtmpfix [see fwtmp(1M)], with the /var/adm/wtmp file as an argument, to correct this situation.

FILES

`/var/adm/wtmp`

SEE ALSO

`acctcom`(1), `login`(1), `acct`(1M), `acctcms`(1M), `acctmerg`(1M), `acctprc`(1M), `acctsh`(1M), `fwtmp`(1M), `init`(1M), `runacct`(1M), `acct`(2), `utmp`(4).

System Administrator's Reference Manual

acctmerg(1M)

NAME

`acctmerg` - merge or add total accounting files

SYNOPSIS

`/usr/lib/acct/acctmerg` [`–aiptuv`] [*file*] ...

DESCRIPTION

`acctmerg` reads its standard input and up to nine additional files, all in binary format or an ASCII version thereof. It merges these inputs by adding records whose keys (normally user ID and name) are identical, and expects the inputs to be sorted on those keys.

OPTIONS

`–a` Produce output in ASCII version from the binary format.

`–i` Input files are ASCII versions of the binary format.

`–p` Print input with no processing.

`–t` Produce a single record that totals all input.

`–u` Summarize by user ID, rather than user ID and name.

`–v` Produce output in verbose ASCII format, with more precise notation for floating-point numbers.

EXAMPLES

The following sequence is useful for making "repairs" to any file kept in this format:

`acctmerg –v <` *file1* `>` *file2*

Edit *file2* as desired ...

`acctmerg –i <` *file2* `>` *file1*

SEE ALSO

`acctcom`(1), `acct`(1M), `acctcms`(1M), `acctcon`(1M), `acctprc`(1M), `acctsh`(1M), `fwtmp`(1M), `runacct`(1M), `acct`(2), `utmp`(4).

System Administrator's Reference Manual

acctprc(1M)

NAME

acctprc, acctprc1, acctprc2 - process accounting

SYNOPSIS

/usr/lib/acct/acctprc

/usr/lib/acct/acctprc1 [*ctmp*]

/usr/lib/acct/acctprc2

DESCRIPTION

acctprc reads standard input, in the form described by acct(4) , and converts it to total accounting records. acctprc divides CPU time into prime time and non-prime time and determines mean memory size (in memory segment units). acctprc then summarizes the binary records, according to user IDs, and adds login names corresponding to the user IDs. The summarized records are then written to standard output. acctprc1 reads input in the form described by acct(4) , adds login names corresponding to user IDs, then writes for each process an ASCII line giving user ID, login name, prime CPU time (tics), non-prime CPU time (tics), and mean memory size (in memory segment units). If *ctmp* is given, it is expected to contain a list of login sessions sorted by user ID and login name. If this file is not supplied, it obtains login names from the password file, just as acctprc does. The information in *ctmp* helps it distinguish between different login names sharing the same user ID.

From standard input, acctprc2 reads records in the form written by acctprc1, summarizes them according to user ID and name, then writes the sorted summaries to standard output as total accounting records.

EXAMPLES

The acctprc command is typically used as shown below:

```
acctprc < /var/adm/pacct > ptacct
```

The acctprc1 and acctprc2 commands are typically used as shown below:

```
acctprc1 ctmp < /var/adm/pacct | acctprc2 > ptacct
```

Example of generating the *ctmp* file:

```
/usr/lib/acct/acctcon1 </var/adm/wtmp | sort 1n +2 >ctmp
```

NOTES

acctprc1 can only distinguish between login names that share user IDs for commands that run normally.

acctprc does not distinguish between users with identical user IDs.

A memory segment of the mean memory size is a unit of measure for the number of bytes in a logical memory segment on a particular processor.

FILES

/etc/passwd

SEE ALSO

acctcom(1) , acct(1M) , acctcms(1M) , acctcon(1M) , acctmerg(1M) , acctsh(1M) , cron(1M) , fwtmp(1M) , runacct(1M) , acct(2) , acct(4) , utmp(4) .

System Administrator's Reference Manual

acctsh(1M)

NAME

acctsh:
chargefee, ckpacct, dodisk, lastlogin, monacct, nulladm, prctmp, prdaily, prtacct, runacct, shutacct, startup, turnacct - shellprocedures for accounting

SYNOPSIS

/usr/lib/acct/chargefee *login-name number*

/usr/lib/acct/ckpacct [*blocks*]

/usr/lib/acct/dodisk [–o] [*files ...*]

/usr/lib/acct/lastlogin

/usr/lib/acct/monacct *number*

/usr/lib/acct/nulladm *file*

/usr/lib/acct/prctmp

/usr/lib/acct/prdaily [–l] [–c] [*mmdd*]

/usr/lib/acct/prtacct *file* ["*heading*"]

/usr/lib/acct/runacct [*mmdd*] [*mmdd state*]

/usr/lib/acct/shutacct ["*reason*"]

/usr/lib/acct/startup

/usr/lib/acct/turnacct on | off | switch

DESCRIPTION

chargefee can be invoked to charge a *number* of units to *login-name*. A record is written to /var/adm/fee, to be merged with other accounting records by runacct.

ckpacct should be initiated via cron(1M) to periodically check the size of /var/adm/pacct. If the size exceeds *blocks*, 1000 by default, turnacct will be invoked with argument *switch*. If the number of free disk blocks in the /var file system falls below 500, ckpacct will automatically turn off the collection of process accounting records via the *off* argument to turnacct. When at least 500 blocks are restored, the accounting will be activated again on the next invocation of ckpacct. This feature is sensitive to the frequency at which ckpacct is executed, usually by cron.

dodisk should be invoked by cron to perform the disk accounting functions. If the –o flag is used, it will use acctdusg [see acct(1M)] to do a slower version of disk accounting by login directory. *files* specifies the one or more filesystem names where disk accounting will be done. If *files* are used, disk accounting will be done on these filesystems only. If the –o flag is used, *files* should be mount points of mounted filesystems. If the –o option is omitted, *files* should be the special file names of mountable filesystems.

lastlogin is invoked by runacct to update /var/adm/acct/sum/loginlog, which shows the last date on which each person logged in.

monacct should be invoked once each month or each accounting period. *number* indicates which month or period it is. If *number* is not given, it defaults to the current month (01–12). This default is useful if monacct is to executed via cron(1M) on the first day of each month. monacct creates summary files in /var/adm/acct/fiscal and restarts the summary files in /var/adm/acct/sum.

nulladm creates *file* with mode 664 and ensures that owner and group are adm. It is called by various accounting shell procedures.

prctmp can be used to print the session record file (normally /var/adm/acct/nite/ctmp created by acctcon1 [see acctcon(1M)].

prdaily is invoked by runacct to format a report of the previous day's accounting data. The report resides in /var/adm/acct/sum/rprt/*mmdd* where *mmdd* is the month and day of the report. The current daily

accounting reports may be printed by typing `prdaily`. Previous days' accounting reports can be printed by using the *mmdd* option and specifying the exact report date desired. The `-l` flag prints a report of exceptional usage by login id for the specified date. Previous daily reports are cleaned up and therefore inaccessible after each invocation of `monacct`. The `-c` flag prints a report of exceptional resource usage by command, and may be used on current day's accounting data only.

`prtacct` can be used to format and print any total accounting (binary) file.

`runacct` performs the accumulation of connect, process, fee, and disk accounting on a daily basis. It also creates summaries of command usage. For more information, see `runacct`(1M) .

`shutacct` is invoked during a system shutdown to turn process accounting off and append a "reason" record to `/var/adm/wtmp`.

`startup` can be invoked when the system is brought to a multiuser state to turn process accounting on.

`turnacct` is an interface to `accton` [see `acct`(1M)] to turn process accounting `on` or `off`. The `switch` argument moves the current `/var/adm/pacct` to the next free name in `/var/adm/pacct`*incr* (where *incr* is a number starting with `1` and incrementing by one for each additional `pacct` file), then turns accounting back on again. This procedure is called by `ckpacct` and thus can be taken care of by the `cron` and used to keep `pacct` to a reasonable size. `shutacct` uses `turnacct` to stop process accounting. `startup` uses `turnacct` to start process accounting.

FILES

`/var/adm/fee`
: accumulator for fees

`/var/adm/pacct`
: current file for per-process accounting

`/var/adm/pacct`*incr*
: used if `pacct` gets large and during execution of daily accounting procedure

`/var/adm/wtmp`
: login/logoff summary

`/usr/lib/acct/ptelus.awk`
: contains the limits for exceptional usage by login ID

`/usr/lib/acct/ptecms.awk`
: contains the limits for exceptional usage by command name

`/var/adm/acct/nite`
: working directory

`/usr/lib/acct`
: holds all accounting commands listed in section 1M

`/var/adm/acct/sum`
: summary directory contains information for `monacct`

`var/adm/acct/fiscal`
: fiscal reports directory

SEE ALSO

`acctcom`(1) , `acct`(1M) , `acctcms`(1M) , `acctcon`(1M) , `acctmerg`(1M) , `acctprc`(1M) , `cron`(1M) , `fwtmp`(1M) , `runacct`(1M) , `acct`(2) , `acct`(4) , `utmp`(4) .

System Administrator's Reference Manual

act_ctlr(1M)
RM400 only

NAME

`act_ctlr, clear_ctlr_attr -` activate/deactivate controllerattributes

SYNOPSIS

`act_ctlr –O|I|E|P –s` *slot_nr* [`–n` *channel*]

`clear_ctlr_attr –O|I|E|P` [`–A`*attr*] `–s` *slot_nr* [`–n` *channel*]

DESCRIPTION

The `act_ctlr` command first checks whether a command has been specified in the `ctlr_defect.tab` table for this board type with which the board must be set to `OK`. If this is the case, the command is executed.

The command `act_ctlr` then deletes the `DEFECT` flag in the configuration object `HW_CONFIG`.

With the command `clear_ctlr_attr`, it is also possible to specify the flag to be processed by the command (1 = `DEFECT`, 2 = `ERROR`). If 1 (`DEFECT`) is specified as the flag, the command is identical to the command `act_ctlr`. If 2 (`ERROR`) is specified as the flag, only the `ERROR` flag in `HW_CONFIG` is deleted. Otherwise, no actions are executed.

OPTIONS

`–O` Onboard bus.

`–I` ISA bus.

`–E` EISA bus.

`–P` PCI bus.

`–A` *attr*
Specification of the attribute number (1 or 2). Default: 2.

`–s` *slot_nr*
Specification of the controller slot number.

`–n` *channel*
Specification of the controller channel number.

Specification of a channel (`–n` *channel*) is ignored at this time, with the exception of CPU boards and MBP-HDs, and the action is executed for the entire board.

RETURN VALUES

If no error occurs, 0 is returned. The other return values have the following meanings:

`1 =` Invalid parameter or option

`2 =` – free –

`3 =` – free –

`4 =` The controller is not available in the system

`5 =` Error when setting the `DEFECT` flag in `HW_CONFIG`

`6 =` Error when setting the `ERROR` flag in `HW_CONFIG`

`7 =` Error when setting the `DEACTIVE` flag in `HW_CONFIG`

`8 =` Error when deleting the `DEFECT` flag in `HW_CONFIG`

`9 =` Error when deleting `ERROR` flag in `HW_CONFIG`

`10 =` Error when deleting the `DEACTIVE` flag in `HW_CONFIG`

`11 =` Error when deactivating the controller

`12 =` Error when activating the controller

FILES

`/usr/bin/lar/ctlr_deact.tab`
`/usr/bin/lar/ctlr_defect.tab`
`/etc/init.d/lar_rc`
`/var/lar/hwconfig`

SEE ALSO

`deact_ctlr(1M)`, `act_dev(1M)`, `deact_dev(1M)`, `lar(7)`, `olr(7)`.

System Administrator's Reference Manual

act_dev(1M)

NAME

`act_dev, clear_dev_attr -` activate/delete SCSI device attributes

SYNOPSIS

`act_dev` *device_name*

`clear_dev_attr` [`–A` *attr*] *device_name*

DESCRIPTION

The `act_dev` command first deletes the `DEFECT` flag in the configuration object for the specified SCSI device. It then checks whether the device is contained in the system configuration (`autoconf`). If this is the case, an attempt is made to reconfigure the device in the system using the `changedev –a` command.

In the case of the `clear_dev_attr` command, the flag to be processed by the command can also be specified (1 = `DEFECT`, 2 = `ERROR`). If 1 (`DEFECT`) is specified as the flag, the command is identical to the `act_dev` command. If 2 (`ERROR`) is specified as the flag, only the `ERROR` flag is deleted in `HW_CONFIG`. Otherwise, no actions are executed.

OPTIONS

`–A` *attr*
: Specifies the attribute number (1 or 2). Default: 2.

device_name
: Name of the device as it is specified using `autoconf –l` (e.g. `ios0/sdisk042`).

RETURN VALUES

If no error occurs, 0 is returned. The other return values have the following meanings:

`1 =` Invalid parameter or option

`2 =` Error processing `HW_CONFIG`

`3 =` Error in `changedev –a`

FILES

`/sbin/scsilar_rc`

`/var/lar/hwconfig`

SEE ALSO

`deact_dev`(1M), `reset_lar`(1M), `set_lar`(1M), `lar`(7), `olr`(7), `autoconf`(8), `changedev`(8).

System Administrator's Reference Manual

add_application(1M)

NAME

`add_application -` add a third-party Web application to WebSysAdmin

SYNOPSIS

`add_application` `–t` *tag* `–n` *name* `–u` *url* `–i` *icon_url*

DESCRIPTION

The `add_application` command adds the information required to start a third-party Web application to the `WSAConfig` file. Based on this information, the third-party Web application can be started from the WebSysAdmin start page.

All of the above-mentioned parameters must be given and possibly quoted ("").

OPTIONS

`–t` *tag*

ID for the Web application in the WebSysAdmin configuration file.

Note:

The ID should be entered with uppercase letters.

`–n` *name*

Name of the application to be added; shown in the bottom line of the application-icon (please do not use more than 8 characters).

`–u` *url*

URL for locating and starting the Web application.

`–i` *icon_url*

URL of the 64x64 pixel icon (gif-format); shown on the WebSysAdmin start page.

Note:

This URL must reference the same server where WebSysAdmin was started.

SEE ALSO

`wsa`(1M) .

System Administrator's Reference Manual

adduser(1M)

NAME

`adduser` - add a new user login to the system

SYNOPSIS

`/sbin/adduser` *login fullname userid home_directory* [*privileges*]

DESCRIPTION

The `adduser` command adds a user to the system, by creating an entry for the user in the `/etc/passwd` and `/etc/shadow` files, setting up the user's home directory, creating a default `.profile` file (if applicable), and setting up an initial password for the user.

The values of *login*, *fullname*, *userid* and *home_directory* are all entered into the `/etc/passwd` file. *login* is the user's login name which must be unique to the system, contain a maximum of eight characters, and consist of numbers and lowercase letters. *fullname* is the user's full name which can be a maximum of 25 characters long, and can consist of letters, numbers and periods. *userid* is the user's unique ID number which must be within the range of 100 to 50000. *home_directory* is the user's home directory. *home_directory* must be unique to the system, contain 14 characters or less, and consist of letters, numbers, and periods. The `adduser` command also sets a default group ID of 1 for the user in the `/etc/passwd` file. The password information is entered in `/etc/shadow`.

If the value of the variable *privileges* is `Yes`, the user will be given system administration privileges. If this option is not used, or any value other than `Yes` is given, no privileges will be assigned.

NOTES

Recommend using `useradd(1M)` instead of `adduser`.

FILES

`/etc/passwd`
`/etc/shadow`
`/etc/.useradm`
`/etc/stdprofile`

SEE ALSO

`deluser(1M)`, `passmgmt(1M)`, `useradd(1M)`.

System Administrator's Reference Manual

autopush(1M)

NAME

`autopush` - configure lists of automatically pushed STREAMSmodules

SYNOPSIS

`autopush –f` *file*

`autopush –r –M` *major* `–m` *minor*

`autopush –g –M` *major* `–m` *minor*

DESCRIPTION

This command allows one to configure the list of modules to be automatically pushed onto the stream when a device is opened. It can also be used to remove a previous setting or get information on a setting.

OPTIONS

`–f` This option sets up the `autopush` configuration for each driver according to the information stored in the specified file. An `autopush` file consists of lines of at least four fields each where the fields are separated by a space as shown below:

maj_ min_ last_min_ mod1 mod2 ... modn

The first three fields are integers that specify the major device number, minor device number, and last minor device number. The fields following represent the names of modules. If *min_* is –1, then all minor devices of a major driver specified by *maj_* are configured and the value for *last_min_* is ignored. If *last_min_* is 0, then only a single minor device is configured. To configure a range of minor devices for a particular major, *min_* must be less than *last_min_*.

The last fields of a line in the `autopush` file represent the list of module names where each is separated by a space. The maximum number of modules that can be automatically pushed on a stream is defined to be eight. The modules are pushed in the order they are specified. Comment lines start with a # sign.

`–r` This option removes the previous configuration setting of the particular *major* and *minor* device number specified with the `–M` and `–m` options respectively. If the values of *major* and *minor* correspond to a setting of a range of minor devices, where *minor* matches the first minor device number in the range, the configuration would be removed for the entire range.

`–g` This option gets the current configuration setting of a particular *major* and *minor* device number specified with the `–M` and `–m` options respectively. It will also return the starting minor device number if the request corresponds to a setting of a range (as described with the `–f` option).

SEE ALSO

`streamio(7)`.

brc(1M) - cron(1M)

System Administrator's Reference Manual

brc(1M)

NAME

`brc, bcheckrc` - system initialization procedures

SYNOPSIS

`/etc/brc`

`/sbin/bcheckrc`

DESCRIPTION

These shell procedures are executed via entries in `/etc/inittab` by `init` whenever the system is booted.

The `brc` script performs operations necessary to initialize the I/O subsystem and the virtual disks. A verbose mode also prints out the current hardware system configuration.

First, the `bcheckrc` procedure checks the status of the `root` file system. If the `root` file system is found to be bad, `bcheckrc` repairs it.

Then, `bckeckrc` mounts the `/stand`, `/proc`, and `/var` (if it exists) file systems (`/var` may exist as a directory in the `root` file system, or as a separate file system).

When these two procedures have been executed, `init` checks for the `initdefault` value in `/etc/inittab`. This tells `init` in which run level to place the system. If, for example, `initdefault` is set to `2`, the system will be placed in the multiuser state via the `rc2` procedure.

Note that `bcheckrc` should always be executed before `brc`. Also, these shell procedures may be used for several run-level states.

SEE ALSO

`fsck`(1M), `init`(1M), `rc2`(1M), `shutdown`(1M), `inittab`(4), `mnttab`(4).

System Administrator's Reference Manual

captoinfo(1M)

NAME

`captoinfo -` convert a termcap description into a terminfo description

SYNOPSIS

`captoinfo [ –v ... ] [ –V ] [ –1 ] [ –w` *width* `]` *file ...*

DESCRIPTION

`captoinfo` looks in *file* for `termcap` descriptions. For each one found, an equivalent `terminfo` description is written to standard output, together with any comments found. A description which is expressed as relative to another description (as specified in the `termcap tc = field`) will be reduced to the minimum superset before being output.

If no *file* is given, the `TERMCAP` environment variable is used for the filename or entry. If `TERMCAP` is a full pathname to a file, only the terminal whose name is specified in the environment variable `TERM` is extracted from that file. If the environment variable `TERMCAP` is not set, then the `/usr/share/lib/termcap` file is read.

OPTIONS

`–v` Prints tracing information to standard error as the program runs. More detailed information is printed if additional `–v` options are specified.

`–V` Prints the version of the program in use to standard error and exits.

`–1` Forces the fields to print one to a line. Otherwise, several fields are printed to a line to a maximum width of 60 characters.

`–w` Changes the output to *width* characters.

NOTES

`captoinfo` should be used to convert `termcap` entries to `terminfo` entries since the `termcap` database may not be supplied in future releases.

FILES

`/usr/share/lib/terminfo/?/*`
Compiled terminal description database

SEE ALSO

`infocmp`(1M), `curses`(3X), `terminfo`(4).

System Administrator's Reference Manual

catman(1M-ucb)
BSD Compatibility Package

NAME

`catman` - create the cat files for the online man pages

SYNOPSIS

`/usr/ucb/catman` [`–nptw`] [`–M` *directory*] [`–T tmac.an`] [*sections*]

DESCRIPTION

The `catman` commands create the preformatted versions of the online man pages from the `nroff(1M)` input files. Each man page is examined and those whose preformatted versions are missing or out of date are recreated. If any changes are made, `catman` recreates the `whatis(1)` database.

If a parameter does not start with a `–`, it is taken to be a list of manual sections to look in. For example

```
catman 123
```

only updates manual sections `1`, `2`, and `3`.

OPTIONS

`–n` Does not (re)create the `whatis(1)` database.

`–p` Prints what would be done instead of doing it.

`–t` Creates `troff`ed entries in the appropriate `fmt` subdirectories instead of `nroff`ing into the `cat` subdirectories.

`–w` Only creates the `whatis(1)` database. No manual reformatting is done.

`–M` Updates man pages located in the specified *directory* (`/usr/share/man` by default).

`–T` Uses `tmac.an` in place of the standard man page macros.

DIAGNOSTICS

`man?/xxx.? (.so'ed from man?/yyy.?): No such file or directory`
The file outside the parentheses is missing, and is referred to by the file inside them.

`target of .so in man?/xxx.? must be relative to /usr/man`
`catman` only allows references to filenames that are relative to the `/usr/share/man` directory.

`opendir:man?: No such file or directory`
A harmless warning message indicating that one of the directories `catman` normally looks for is missing.

`*.*: No such file or directory`
A harmless warning message indicating `catman` came across an empty directory.

ENVIRONMENT

`TROFF`
The name of the formatter to use when the `–t` flag is given. If not set, `troff(1M)` is used.

CAVEAT

This `catman` command will not create preformatted DC/OSx man pages. It is included for users who have BSD man pages.

FILES

`/usr/share/man`
Default manual directory location

`/usr/share/man/man?/*.*`

Raw (`nroff` input) manual sections

`/usr/share/man/cat?/*.*`
Preformatted `nroff`ed man pages

`/usr/share/man/fmt?/*.*`
Preformatted `troff`ed man pages

`/usr/share/man/whatis`
`whatis(1)` database location

`/usr/ucblib/makewhatis`
Command script to make `whatis(1)` database

SEE ALSO

`man(1)`, `whatis(1)`, `man(1-ucb)`, `nroff(1M)`, `troff(1M)`, `man(5)`, `troff(5)`.

System Administrator's Reference Manual

catman(1M)

NAME

`catman -` create the `cat` files for the online man pages

SYNOPSIS

`/usr/bin/catman` [`–v`] [`–f`] [`–n`] [`–m` *man_path_dot*] [`–u`] [`–d` *alt_mandb*]

DESCRIPTION

The `catman` commands create the preformatted versions of the online man page from the `nroff(1M)` input files. Each man page is examined and those whose preformatted versions are missing or out of date are recreated. If any changes are made, `catman` recreates the `mandb` database.

OPTIONS

`–v` The `–v` option turns on debug.

`–f` The `–f` option forces `cat` pages to be built, even if the `cat` page is newer.

`–n` If the `–n` option is set, German umlauts in the `nroff` source files are replaced by ae, oe, ue, etc.

`–m` The *man_path_dot* argument specifies the directory under which the man pages to be formatted can be found. If the `–m` option is not specified, the search for the man pages to be formatted begins from the current directory.

`–u` If the `–u` option is given, `catman` will only attempt to rebuild all `mandb` files along the `$MANPATH`.

`–d` If the `–d` option is given, `catman` will *NOT* attempt to rebuild any `mandb` file along the `$MANPATH`. It will use the *alt_mandb* to determine what should be built, and how to build it (`neqn`, `tbl`, ...).

The directory search for man pages is controlled by the `$MANPATH` environment variable, or its default:

`/usr/share/man/mrd:/usr/share/man:/usr/share/man/mrd/$LANG`

One of 2 methods will be used to look for man pages, based on whether or not the top level directory contains a `.mandb` directory.

If no `.mandb` subdirectory is found, man pages are assumed to be all files under subdirectories beginning with `man` or ending with `_man`.

If a `.mandb` subdirectory is found, then the file `mandb` will be used to locate man pages. If `mandb` does not exist, or is older than the modification time of `.mandb`, then `mandb` will be constructed from all files under `.mandb`. The `mandb` file is used to locate man pages.

NOTE:

Even though *alt_mandb* may be an absolute path, the paths within it must be relative to the `$MANPATH` location. The `$MANPATH` location must have a `.mandb` directory, though its contents are not used explicitly.

This utility makes use of both `cat` pages from man pages. A man page is an `nroff` file written with the man macro set. A `cat` page is formed by formatting a man page with `nroff`.

A `cat` page path is of the form:

`./catman/`*some_path*`/`*filename*`.Z`

corresponds to a man page of the form:

./some_path/filename

where "`.`" is the top level search directory. If the `–m` option *man_path_dot* is used, then the corresponding man page is:

man_path_dot/some_path/filename

By default, all `cat` pages that are older than existing and corresponding man pages are rebuilt. The `–f` option forces all `cat` pages corresponding to existing man pages to be rebuilt.

NOTES

`catman` analyzes the following data from the header of the `nroff` source files to create the `mandb`: *title*, *section*, and *short_comment* [cf. `man(1)`]. For example:

```
.nr X 2
.if \nX=0 .ds x} title section "" ""
.if \nX=1 .ds x} title section ""
.if \nX=2 .ds x} title section "" "\&"
.TH \*(x}
.SH NAMEtitle \- short_comment
```

ENVIRONMENT

`MANPATH`
: Path under which a search is performed for the `mandb` and the preformatted man pages. Default: `/usr/share/man/mrd:/usr/share/man:/usr/share/man/mrd/$LANG`.

`LANG` Environment variable for the language setting. Default: `En_US.ASCII`.

`NROFF`
: The `nroff` formatter used is taken from the `$NROFF` environment variable. If `$NROFF` does not exist, or is empty, then `/usr/ucb/nroff` or `/usr/bin/nroff` (if `/usr/ucb/nroff` does not exist) will be used.

FILES

`/usr/share/man/mrd`
: Default manual directory location

`/usr/share/man/mrd/En_US.ASCII`
: Directory for the English `mandb` and the preformatted man pages

`/usr/share/man/mrd/De_DE.646`
: Directory for the German `mandb` and the preformatted man pages

`/usr/share/man/mrd/$LANG/mandb`
: Path containing the complete `mandb`

`/usr/share/man/mrd/$LANG/.mandb`
: Directory containing the man pages belonging to add-on packages (required for the complete `mandb`)

`/usr/share/man/mrd/$LANG/catman`
: Directory containing the preformatted and compressed man pages (*filename*`.Z`)

SEE ALSO

`man(1)`, `man(1-ucb)`, `neqn(1M)`, `nroff(1M)`, `tbl(1M)`, `man(5)`.

System Administrator's Reference Manual

checkmm(1M)

NAME

`checkmm, checkeq -` check documents formatted with themm macros

SYNOPSIS

`checkmm` [*files*]

`checkeq` [*files*]

DESCRIPTION

`checkmm` stands for "check memorandum macros". Use `checkmm` or `checkeq` to check for syntax errors in files that have been prepared for the `mm`(1M) or `mmt`(1M) command. For example, `checkmm` checks that you have a `.DE` (display end macro) corresponding to every `.DS` (display start macro).

The output for `checkmm` is the number of lines checked, and a list of macros that are unfinished because of missing macros. If you do not include a filename on the command line, `checkmm` takes input from standard input.

DIAGNOSTICS

"`checkmm Cannot open` *filename*"
File(s) is(are) unreadable

The remaining output of the program is diagnostic of the source file.

SEE ALSO

`eqn`(1M), `mm`(1M), `mmt`(1M), `mvt`(1M), `neqn`(1M), `tbl`(1M).

System Administrator's Reference Manual

chroot(1M)

NAME

`chroot` - change root directory for a command

SYNOPSIS

`/usr/sbin/chroot` *newroot command*

DESCRIPTION

`chroot` causes the given command to be executed relative to the new root. The meaning of any initial slashes (`/`) in the path names is changed to *newroot* for the command and any of its child processes. Furthermore, upon execution, the initial working directory is *newroot*.

Notice, however, that if you redirect the output of the command to a file:

`chroot` *newroot command* `>` `x`

will create the file `x` relative to the original root of the command, not the new one.

The new root path name is always relative to the current root: even if a `chroot` is currently in effect, the *newroot* argument is relative to the current root of the running process.

This command can only be executed by the superuser.

LOCALE

The `LC_MESSAGES` environment variable governs the language in which message texts are displayed.

If `LC_MESSAGES` is undefined or is defined as the null string, it defaults to the value of `LANG`. If `LANG` is likewise undefined or null, the system acts as if it were not internationalized.

If any of the locale variables has an invalid value, the system acts as if none of the variables was set.

The `LC_ALL` environment variable governs the entire locale. `LC_ALL` takes precedence over all the other environment variables that affect internationalization.

NOTES

One should exercise extreme caution when referencing device files in the new root file system.

SEE ALSO

`cd(1)` , `chroot(2)` .

System Administrator's Reference Manual

chrtbl(1M)

NAME

chrtbl - generate character classification and conversion tables

SYNOPSIS

chrtbl [*file*]

DESCRIPTION

The chrtbl command creates two tables containing information on character classification, upper/lowercase conversion, character-set width, and numeric formatting. One table is an array of (257* 2) + 7 bytes which is encoded to enable a table lookup to be used to determine the character classification of a character, convert a character [see ctype(3C)], and find the byte and screen width of a character in one of the supplementary code sets. The other table contains information about the format of non-monetary numeric quantities: the first byte specifies the decimal delimiter; the second byte specifies the thousands delimiter; and the remaining bytes comprise a null-terminated string indicating the grouping (each element of the string is taken as an integer that indicates the number of digits that comprise the current group in a formatted non-monetary numeric quantity).

chrtbl reads the user-defined character classification and conversion information from *file* and creates three output files in the current directory. To construct *file*, use the file supplied in /usr/lib/locale/C/chrtbl_C as a starting point. You may add entries, but do not change the original values supplied with the system. For example, for other locales you may wish to add eight-bit entries to the ASCII definitions provided in this file.

One output file, ctype.c (a C-language source file), contains a (257* 2)+7-byte array generated from processing the information from *file*. You should review the content of ctype.c to verify that the array is set up as you had planned. (In addition, an application program could use ctype.c.) The first 257 bytes of the array in ctype.c are used for character classification. The characters used for initializing these bytes of the array represent character classifications that are defined in /usr/include/ctype.h; for example, _S means a character is a spacing character and _U|_L means the character is uppercase or lowercase. The second 257 bytes of the array are used for character conversion. These bytes of the array are initialized so that characters for which you do not provide conversion information will be converted to themselves. When you do provide conversion information, the first value of the pair is stored where the second one would be stored normally, and vice versa; for example, if you provide <0x41□0x61>, then 0x61 is stored where 0x41 would be stored normally, and 0x61 is stored where 0x41 would be stored normally. The last 7 bytes are used for character width information for up to three supplementary code sets.

The second output file (a data file) contains the same information, but is structured for efficient use by the character classification and conversion routines [see ctype(3C)]. The name of this output file is the value you assign to the keyword LC_CTYPE read in from *file*. Before this file can be used by the character classification and conversion routines, it must be installed in the /usr/lib/locale/*locale* directory with the name LC_CTYPE by someone who is superuser or a member of group bin. This file must be readable by user, group, and other; no other permissions should be set. To use the character classification and conversion tables in this file, set the LC_CTYPE environment variable appropriately [see environ(5) or setlocale(3C)].

The third output file (a data file) is created only if numeric formatting information is specified in the input file. The name of this output file is the value you assign to the keyword LC_NUMERIC read in from *file*. Before this file can be used, it must be installed in the /usr/lib/locale/*locale* directory with the name LC_NUMERIC by a superuser or a member of group bin. This file must be readable by user, group, and other; no other permissions should be set. To use the numeric formatting information in this file, set the LC_NUMERIC environment variable appropriately [see environ(5) or setlocale(3C)].

The name of the locale where you install the files LC_CTYPE and LC_NUMERIC should correspond to the conventions defined in *file*. For example, if French conventions were defined, and the name for the French

locale on your system is `french`, then you should install the files in `/usr/lib/locale/french`.

If no input file is given, or if the argument "–" is encountered, `chrtbl` reads from standard input.

The syntax of *file* allows the user to define the names of the data files created by `chrtbl`, the assignment of characters to character classifications, the relationship between uppercase and lowercase letters, byte and screen widths for up to three supplementary code sets, and three items of numeric formatting information: the decimal delimiter, the thousands delimiter and the grouping. The keywords recognized by `chrtbl` are:

`LC_CTYPE`
: Name of the data file created by `chrtbl` to contain character classification, conversion, and width information.

`isupper`
: Character codes to be classified as uppercase letters.

`islower`
: Character codes to be classified as lowercase letters.

`isdigit`
: Character codes to be classified as numeric.

`isspace`
: Character codes to be classified as spacing (delimiter) characters.

`ispunct`
: Character codes to be classified as punctuation characters.

`iscntrl`
: Character codes to be classified as control characters.

`isblank`
: Character code for the space character.

`isxdigit`
: Character codes to be classified as hexadecimal digits.

`ul`
: Relationship between uppercase and lowercase characters.

`cswidth`
: Byte and screen width information (by default, each is one character wide).

`LC_NUMERIC`
: Name of the data file created by `chrtbl` to contain numeric formatting information.

`decimal_point`
: Decimal delimiter.

`thousands_sep`
: Thousands delimiter.

`grouping`
: String in which each element is taken as an integer that indicates the number of digits that comprise the current group in a formatted non-monetary numeric quantity.

Any lines with the number sign (`#`) in the first column are treated as comment and are ignored. Blank lines are also ignored.

Characters for `isupper`, `islower`, `isdigit`, `isspace`, `ispunct`, `iscntrl`, `isblank`, `isxdigit`, and `ul` can be represented as a hexadecimal or octal constant (for example, the letter `a` can be represented as `0x61` in hexadecimal or `0141` in octal). Hexadecimal and octal constants may be separated by one or more space and/or tab characters.

The dash character (`–`) may be used to indicate a range of consecutive numbers. Zero or more space characters may be used for separating the dash character from the numbers.

The backslash character (`\`) is used for line continuation. Only a carriage return is permitted after the backslash character.

The relationship between uppercase and lowercase letters (`ul`) is expressed as ordered pairs of octal or hexadecimal constants: <*uppercase_character lowercase_character*>. These two constants may be separated by

one or more space characters. Zero or more space characters may be used for separating the angle brackets (<>) from the numbers.

The following is the format of an input specification for `cswidth`: *n1*:*s1*,*n2*:*s2*,*n3*:*s3* where,

n1 byte width for supplementary code set 1, required

s1 screen width for supplementary code set 1

n2 byte width for supplementary code set 2

s2 screen width for supplementary code set 2

n3 byte width for supplementary code set 3

s3 screen width for supplementary code set 3

`decimal_point` and `thousands_sep` are specified by a single character that gives the delimiter. `grouping` is specified by a quoted string in which each member may be in octal or hex representation. For example, `\3` or `\x3` could be used to set the value of a member of the string to 3.

EXAMPLES

The following is an example of an input file used to create the USA-ENGLISH code set definition table in a file named `usa` and the non-monetary numeric formatting information in a file name `num-usa`.

```
LC_CTYPE  usa□
isupper   0x41 - 0x5a□
islower   0x61 - 0x7a□
isdigit   0x30 - 0x39□
isspace   0x20 0x9 - 0xd□
ispunct   0x21 - 0x2f   0x3a - 0x40   0x5b - 0x60   0x7b - 0x7e□
iscntrl   0x0 - 0x1f 0x7f□
isblank   0x20□
isxdigit  0x30 - 0x39   0x61 - 0x66   0x41 - 0x46□
ul       <0x41 0x61> <0x42 0x62> <0x43 0x63>  \□
          <0x44 0x64> <0x45 0x65> <0x46 0x66>  \□
          <0x47 0x67> <0x48 0x68> <0x49 0x69>  \□
          <0x4a 0x6a> <0x4b 0x6b> <0x4c 0x6c>  \□
          <0x4d 0x6d> <0x4e 0x6e> <0x4f 0x6f>  \□

       <0x50 0x70> <0x51 0x71> <0x52 0x72>  \□
          <0x53 0x73> <0x54 0x74> <0x55 0x75>  \□
          <0x56 0x76> <0x57 0x77> <0x58 0x78>  \□
          <0x59 0x79> <0x5a 0x7a>□
cswidth         1:1,0:0,0:0□
LC_NUMERIC      num_usa□
decimal_point   .□
thousands_sep   ,□
grouping        "\3"
```

LOCALE

The `LC_MESSAGES` environment variable governs the language in which message texts are displayed.

If `LC_MESSAGES` is undefined or is defined as the null string, it defaults to the value of `LANG`. If `LANG` is likewise undefined or null, the system acts as if it were not internationalized.

If any of the locale variables has an invalid value, the system acts as if none of the variables was set.

The `LC_ALL` environment variable governs the entire locale. `LC_ALL` takes precedence over all the other environment variables that affect internationalization.

DIAGNOSTICS

The error messages produced by `chrtbl` are intended to be self-explanatory. They indicate errors in the command line or syntactic errors encountered within the input file.

WARNING

Changing the files in `/usr/lib/locale/C` will cause the system to behave unpredictably.

FILES

`/usr/lib/locale/`*locale*`/`LC_CTYPE
: data files containing character classification, conversion, and character-set width information created by `chrtbl`

`/usr/lib/locale/`*locale*`/`LC_NUMERIC
: data files containing numeric formatting information created by `chrtbl`

`/usr/include/ctype.h`
: header file containing information used by character classification and conversion routines

`/usr/lib/locale/C/chrtbl_C`
: input file used to construct `LC_CTYPE` and `LC_NUMERIC` in the default locale

SEE ALSO

`ctype`(3C), `setlocale`(3C), `environ`(5).

System Administrator's Reference Manual

ckbinarsys(1M)

NAME

ckbinarsys - determine whether remote system can accept binarymessages

SYNOPSIS

/usr/lib/mail/surrcmd/ckbinarsys [–S] –s *remote_system_name* –t *content_type*

DESCRIPTION

Because rmail can transport binary data, it may be important to determine whether a particular remote system (typically the next hop) can handle binary data via the chosen transport layer agent (uux, SMTP, etc.).

ckbinarsys consults the /etc/mail/binarsys file for information on a specific remote system. ckbinarsys returns its results via an appropriate exit code. An exit code of zero implies that it is OK to send a message with the indicated content type to the system specified. An exit code other than zero indicates that the remote system cannot properly handle messages with binary content.

The absence of the binarsys file will cause ckbinarsys to exit with a non-zero exit code.

OPTIONS

–s *remote_system_name*
: Name of remote system to look up in /etc/mail/binarsys

–t *content_type*
: Content type of message to be sent. When invoked by rmail, this will be one of two strings: text or binary, as determined by mail independent of any Content-Type: header lines that may be present within the message header. All other arguments are treated as equivalent to binary.

–S
: Normally, ckbinarsys will print a message (if the binary mail is rejected) which would be suitable for rmail to return in the negative acknowledgement mail. When –S is specified, no message will be printed.

FILES

/etc/mail/binarsys
/usr/lib/mail/surrcmd/ckbinarsys

SEE ALSO

mail(1), uux(1), binarsys(4), mailsurr(4).

System Administrator's Reference Manual

ckdate(1M)

NAME

`ckdate, errdate, helpdate, valdate -` prompts forand validates a date

SYNOPSIS

`/usr/bin/ckdate` [`–Q`] [`–W` *width*] [`–f`*format*] [`–d` *default*] [`–h` *help*] [`–e` *error*] [`–p` *prompt*] [`–k` *pid* [`–s` *signal*]]

`/usr/sadm/bin/errdate` [`–W`] [`–e` *error*] [`–f`*format*]

`/usr/sadm/bin/helpdate` [`–W`] [`–h` *help*] [`–f`*format*]

`/usr/sadm/bin/valdate` [`–f` *format*] *input*

DESCRIPTION

`ckdate` prompts a user and validates the response. It defines, among other things, a prompt message whose response should be a date, text for help and error messages, and a default value (which will be returned if the user responds with a carriage return). The user response must match the defined format for a date.

All messages are limited in length to 70 characters and are formatted automatically. Any white space used in the definition (including newline) is stripped. The `–W` option cancels the automatic formatting. When a tilde is placed at the beginning or end of a message definition, the default text will be inserted at that point, allowing both custom text and the default text to be displayed.

If the prompt, help or error message is not defined, the default message (as defined under **NOTES**) will be displayed.

Three visual tool modules are linked to the `ckdate` command. They are `errdate` (which formats and displays an error message), `helpdate` (which formats and displays a help message), and `valdate` (which validates a response). These modules should be used in conjunction with FML objects. In this instance, the FML object defines the prompt. When *format* is defined in the `errdate` and `helpdate` modules, the messages will describe the expected format.

OPTIONS

`–Q` Specifies that quit will not be allowed as a valid response.

`–W` *width*
: Specifies that prompt, help and error messages will be formatted to a line length of *width*.

`–f` *format*
: Specifies the format against which the input will be verified. Possible formats and their definitions are:

  `%b` Abbreviated month name (jan, feb, mar); cf. `LC_TIME(4)` .

  `%h` Same as `%b`.

  `%B` Full month name.

  `%d` Day of month (01 – 31).

  `%D` Date as `%m/%d/%y` (the default format); cf. `date(1)` .

  `%e` Day of month (1 – 31; single digits are preceded by a blank).

  `%m` Month number (01 – 12).

  `%y` Year within century (e.g. 89).

  If the two-digit year notation is in the range `70` to `99`, the 20th century is assumed; if it is in the range `00` to `37`, the 21st century is assumed. All other entries are rejected.

  `%Y` Year as *CCYY* (e.g. 1989).

`–d` *default*

Defines the default value as *default*. The default does not have to meet the format criteria.

`-h` *help*
Defines the help messages as *help*.

`-e` *error*
Defines the error message as *error*.

`-p` *prompt*
Defines the prompt message as *prompt*.

`-k` *pid*
Specifies that process ID *pid* is to be sent a signal if the user chooses to abort.

`-s` *signal*
Specifies that the process ID *pid* defined with the `-k` option is to be sent signal `signal` when quit is chosen. If no signal is specified, `SIGTERM` is used.

input
Input to be verified against format criteria.

EXIT CODES

`0` = Successful execution
`1` = EOF on input
`2` = Usage error
`3` = User termination (quit)
`4` = Garbled format argument

NOTES

The default prompt for `ckdate` is:

```
Enter the date [?,q]:
```

The default error message is:

```
ERROR: Please enter a date. Format is: <format>.
```

The default help message is:

```
Please enter a date. Format is: <format>.
```

When the quit option is chosen (and allowed), `q` is returned along with the return code `3`. The `valdate` module will not produce any output. It returns zero for success and non-zero for failure.

System Administrator's Reference Manual

ckgid(1M)

NAME

`ckgid, errgid, helpgid, valgid -` prompts for andvalidates a group ID

SYNOPSIS

`/usr/bin/ckgid` [`–Q`] [`–W` *width*] [`–m`] [`–d` *default*] [`–h` *help*] [`–e` *error*] [`–p` *prompt*] [`–k` *pid* [`–s` *signal*]]

`/usr/sadm/bin/errgid` [`–W`] [`–e` *error*]

`/usr/sadm/bin/helpgid` [`–W`] [`–m`] [`–h` *help*]

`/usr/sadm/bin/valgid` *input*

DESCRIPTION

`ckgid` prompts a user and validates the response. It defines, among other things, a prompt message whose response should be an existing group ID, text for help and error messages, and a default value (which will be returned if the user responds with a carriage return).

All messages are limited in length to 70 characters and are formatted automatically. Any white space used in the definition (including newline) is stripped. The `–W` option cancels the automatic formatting. When a tilde is placed at the beginning or end of a message definition, the default text will be inserted at that point, allowing both custom text and the default text to be displayed.

If the prompt, help or error message is not defined, the default message (as defined under **NOTES**) will be displayed.

Three visual tool modules are linked to the `ckgid` command. They are `errgid` (which formats and displays an error message), `helpgid` (which formats and displays a help message), and `valgid` (which validates a response). These modules should be used in conjunction with FML objects. In this instance, the FML object defines the prompt.

OPTIONS

The options and arguments for this command are:

`–Q` Specifies that quit will not be allowed as a valid response.

`–W` *width*
: Specifies that prompt, help, and error messages will be formatted to a line length of *width*.

`–m` Displays a list of all groups when help is requested or when the user makes an error.

`–d` *default*
: Defines the default value as *default*. The default is not validated and so does not have to meet any criteria.

`–h` *help*
: Defines the help messages as *help*.

`–e` *error*
: Defines the error message as *error*.

`–p` *prompt*
: Defines the prompt message as *prompt*.

`–k` *pid*
: Specifies that process ID *pid* is to be sent a signal if the user chooses to abort.

`–s` *signal*
: Specifies that the process ID *pid* defined with the `–k` option is to be sent a signal *signal* when quit is chosen. If no signal is specified, `SIGTERM` is used.

input
: Input to be verified against `/etc/group`.

DIAGNOSTICS

The exit codes for the `ckgid` command are the following:

`0` = Successful execution

`1` = EOT on input

`2` = Usage error

`3` = User termination (quit)

NOTES

The default prompt for `ckgid` is:

```
Enter the name of an existing group [?,q]:
```

The default error message is:

```
ERROR - Please enter the name of an existing group.
```

(If the `–m` option of `ckgid` is used, a list of valid groups is displayed here.)

The default help message is:

```
Please enter an existing group name.
```

(If the `–m` option of `ckgid` is used, a list of valid groups is displayed here.)

When the quit option is chosen (and allowed), `q` is returned along with the return code `3`. The `valgid` module will not produce any output. It returns zero for success and non-zero for failure.

System Administrator's Reference Manual

ckint(1M)

NAME

`ckint, errint, helpint, valint -` display a prompt;verify and return an integer value

SYNOPSIS

`/usr/bin/ckint` [`–Q`] [`–W` *width*] [`–b` *base*] [`–d` *default*] [`–h` *help*] [`–e` *error*] [`–p` *prompt*] [`–k` *pid* [`–s` *signal*]]

`/usr/sadm/bin/errint` [`–W`] [`–b` *base*] [`–e` *error*]

`/usr/sadm/bin/helpint` [`–W`] [`–b` *base*] [`–h` *help*]

`/usr/sadm/bin/valint` [`–b` *base*] *input*

DESCRIPTION

`ckint` prompts a user, then validates the response. It defines, among other things, a prompt message whose response should be an integer, text for help and error messages, and a default value (which will be returned if the user responds with a carriage return).

All messages are limited in length to 70 characters and are formatted automatically. Any white space used in the definition (including newline) is stripped. The `–W` option cancels the automatic formatting. When a tilde is placed at the beginning or end of a message definition, the default text will be inserted at that point, allowing both custom text and the default text to be displayed.

If the prompt, help or error message is not defined, the default message (as defined under **NOTES**) will be displayed.

Three visual tool modules are linked to the `ckint` command. They are `errint` (which formats and displays an error message), `helpint` (which formats and displays a help message), and `valint` (which validates a response). These modules should be used in conjunction with FML objects. In this instance, the FML object defines the prompt. When *base* is defined in the `errint` and `helpint` modules, the messages will include the expected base of the input.

OPTIONS

The options and arguments for this command are:

`–Q` Specifies that quit will not be allowed as a valid response.

`–W` Specifies that prompt, help and error messages will be formatted to a line length of *width*.

`–b` Defines the base for input. Must be 2 to 36, default is 10.

`–d` Defines the default value as *default*. The default is not validated and so does not have to meet any criteria.

`–h` Defines the help messages as *help*.

`–e` Defines the error message as *error*.

`–p` Defines the prompt message as *prompt*.

`–k` Specifies that process ID *pid* is to be sent a signal if the user chooses to abort.

`–s` Specifies that the process ID *pid* defined with the `–k` option is to be sent signal `signal` when quit is chosen. If no signal is specified, `SIGTERM` is used.

input
Input to be verified against *base* criterion.

EXIT CODES

`0` = Successful execution

`1` = EOF on input

`2` = Usage error

`3` = User termination (quit)

NOTES

The default base 10 prompt for `ckint` is:

```
Enter an integer [?,q]:
```

The default base 10 error message is:

```
ERROR - Please enter an integer.
```

The default base 10 help message is:

```
Please enter an integer.
```

The messages are changed from "integer" to "base *base* integer" if the base is set to a number other than 10.

When the quit option is chosen (and allowed), `q` is returned along with the return code `3`. The `valint` module will not produce any output. It returns zero for success and non-zero for failure.

System Administrator's Reference Manual

ckitem(1M)

NAME

ckitem, erritem, helpitem - build a menu; prompt for andreturn a menu item

SYNOPSIS

/usr/bin/ckitem [–Q] [–W *width*] [–uno] [–f *file*] [–l *label*] [[–i *invis*] [, ...]] [–m *max*] [–d *default*] [–h *help*] [–e *error*] [–p *prompt*] [–k *pid* [–s *signal*]] [*choice* [...]]

/usr/sadm/bin/erritem [–W] [–e *error*] [*choice* [...]]

/usr/sadm/bin/helpitem [–W] [–h *help*] [*choice* [...]]

DESCRIPTION

ckitem builds a menu and prompts the user to choose one item from a menu of items. It then verifies the response. Options for this command define, among other things, a prompt message whose response will be a menu item, text for help and error messages, and a default value (which will be returned if the user responds with a carriage return).

Unless otherwise specified, the menu is formatted so that each item precedes a number and the elements comprising an item appear in columns on the screen. The column width is determined by the longest item. The items are arranged alphabetically.

All messages are limited in length to 70 characters and are formatted automatically. Any white space used in the definition (including newline) is stripped. The –W option cancels the automatic formatting. When a tilde is placed at the beginning or end of a message definition, the default text will be inserted at that point, allowing both custom text and the default text to be displayed.

If the prompt, help or error message is not defined, the default message (as defined under **NOTES**) will be displayed.

Two visual tool modules are linked to the ckitem command. They are erritem (which formats and displays an error message) and helpitem (which formats and displays a help message). These modules should be used in conjunction with FML objects. In this instance, the FML object defines the prompt. When *choice* is defined in these modules, the messages will describe the available menu choice (or choices).

OPTIONS

The options and arguments for this command are:

–Q Specifies that quit will not be allowed as a valid response.

–W Specifies that prompt, help and error messages will be formatted to a line length of *width*.

–u Specifies that menu items should be displayed as an unnumbered list.

–n Specifies that menu items should not be displayed in alphabetical order.

–o Specifies that only one menu token will be returned.

–f Defines a file, *file*. which contains a list of menu items to be displayed. (The format of this file is: token<tab>description. Lines beginning with a pound sign (#) are designated as comments and ignored.)

–l Defines a label, *label*, to print above the menu.

–i Defines invisible menu choices (those which will not be printed in the menu). (For example, "all" used as an invisible choice would mean it is a legal option but does not appear in the menu. Any number of invisible choices may be defined.) Invisible choices should be made known to a user either in the prompt or in a help message.

–m Defines the maximum number of menu choices allowed.

–d Defines the default value as *default*. The default is not validated and so does not have to meet any

criteria.

`-h` Defines the help messages as *help*.

`-e` Defines the error message as *error*.

`-p` Defines the prompt message as *prompt*.

`-k` Specifies that the process ID *pid* is to be sent a signal if the user chooses to abort.

`-s` Specifies that process ID *pid* defined with the `-k` option is to be sent signal `signal` when quit is chosen. If no signal is specified, `SIGTERM` is used.

choice
Defines menu items. Items should be separated by white space or newline.

EXIT CODES

`0` =Successful execution

`1` =EOF on input

`2` =Usage error

`3` =User termination (quit)

`4` =No choices from which to choose

NOTES

The user may input the number of the menu item if choices are numbered or as much of the string required for a unique identification of the item. Long menus are paged with 10 items per page.

When menu entries are defined both in a file (by using the `-f` option) and also on the command line, they are usually combined alphabetically. However, if the `-n` option is used to suppress alphabetical ordering, then the entries defined in the file are shown first, followed by the options defined on the command line.

The default prompt for `ckitem` is:

```
Enter selection [?,??,q]:
```

One question mark will give a help message and then redisplay the prompt. Two question marks will give a help message and then redisplay the menu label, the menu and the prompt.

The default error message is:

```
ERROR - Does not match an available menu selection.
Enter one of the following:
-- the number of the menu item you wish to select
-- the token associated withe the menu item
-- partial string which uniquely identifies the token for the menu item
\{\
-- partial string which uniquely identifies the token for the
  menu item
-- ?? to reprint the menu

The default help message is:
Enter one of the following:
-- the number of the menu item you wish to select
-- the token associated with the menu item
\{\
-- partial string which uniquely identifies the token for the
  menu item
-- partial string which uniquely identifies the token for the menu item
-- ?? to reprint the menu

When the quit option is chosen (and allowed), q is
```

returned along with the return code `3`.

System Administrator's Reference Manual

ckkeywd(1M)

NAME

`ckkeywd -` prompt for and validate a keyword

SYNOPSIS

`ckkeywd` [`–Q`] [`–W` *width*] [`–d` *default*] [`–h` *help*] [`–e` *error*] [`–p` *prompt*] [`–k` *pid* [`–s` *signal*]] [*keyword* [...]]

DESCRIPTION

`ckkeywd` prompts a user and validates the response. It defines, among other things, a prompt message whose response should be one of a list of keywords, text for help and error messages, and a default value (which will be returned if the user responds with a carriage return). The answer returned from this command must match one of the defined list of keywords.

All messages are limited in length to 70 characters and are formatted automatically. Any white space used in the definition (including newline) is stripped. The `–W` option cancels the automatic formatting. When a tilde is placed at the beginning or end of a message definition, the default text will be inserted at that point, allowing both custom text and the default text to be displayed.

If the prompt, help or error message is not defined, the default message (as defined under **NOTES**) will be displayed.

OPTIONS

`–Q` Specifies that quit will not be allowed as a valid response.

`–W` Specifies that prompt, help and error messages will be formatted to a line length of *width*.

`–d` Defines the default value as *default*. The default is not validated and so does not have to meet any criteria.

`–h` Defines the help messages as *help*.

`–e` Defines the error message as *error*.

`–p` Defines the prompt message as *prompt*.

`–k` Specifies that process ID *pid* is to be sent a signal if the user chooses to abort.

`–s` Specifies that the process ID *pid* defined with the `–k` option is to be sent signal `signal` when quit is chosen. If no signal is specified, `SIGTERM` is used.

keyword
Defines the keyword, or list of keywords, against which the answer will be verified.

EXIT CODES

`0` =Successful execution

`1` =EOF on input

`2` =Usage error

`3` =User termination (quit)

`4` =No keywords from which to choose

NOTES

The default prompt for `ckkeywd` is:

`Enter selection [`*keyword*`,[...],?,q]:`

The default error message is:

```
ERROR - Does not match any of the valid selections.
```

`Please enter one of the following keywords:`
keyword`[,...]`
The default help message is:
`Please enter one of the following keywords:`
keyword`[,...]`
When the quit option is chosen (and allowed), `q` is returned along with the return code `3`.

System Administrator's Reference Manual

ckpath(1M)

NAME

`ckpath, errpath, helppath, valpath -` display aprompt; verify and return a pathname

SYNOPSIS

`/usr/bin/ckpath` [`–Q`] [`–W` *width*] [`–a|l`] [`–b|c|g|y`] [`–n|[o|z`]] [`–rtwx`] [`–d` *default*] [`–h` *help*] [`–e` *error*] [`–p` *prompt*] [`–k` *pid* [`–s` *signal*]]

`/usr/sadm/bin/errpath` [`–W`] [`–a|l`] [`–b|c|g|y`] [`–n|[o|z`]] [`–rtwx`] [`–e` *error*]

`/usr/sadm/bin/helppath` [`–W`] [`–a|l`] [`–b|c|g|y`] [`–n|[o|z`]] [`–rtwx`] [`–h` *help*]

`/usr/sadm/bin/valpath` [`–a|l`] [`–b|c|g|y`] [`–n|[o|z`]] [`–rtwx`] *input*

DESCRIPTION

`ckpath` prompts a user and validates the response. It defines, among other things, a prompt message whose response should be a pathname, text for help and error messages, and a default value (which will be returned if the user responds with a carriage return).

The pathname must obey the criteria specified by the first group of options. If no criteria is defined, the pathname must be for a normal file that does not yet exist. If neither `–a` (absolute) or `–l` (relative) is given, then either is assumed to be valid.

All messages are limited in length to 70 characters and are formatted automatically. Any white space used in the definition (including newline) is stripped. The `–W` option cancels the automatic formatting. When a tilde is placed at the beginning or end of a message definition, the default text will be inserted at that point, allowing both custom text and the default text to be displayed.

If the prompt, help or error message is not defined, the default message (as defined under **NOTES**) will be displayed.

Three visual tool modules are linked to the `ckpath` command. They are `errpath` (which formats and displays an error message), `helppath` (which formats and displays a help message), and `valpath` (which validates a response). These modules should be used in conjunction with FACE objects. In this instance, the FACE object defines the prompt.

OPTIONS

The options and arguments for this command are:

`–Q` Specifies that quit will not be allowed as a valid response.

`–W` Specifies that prompt, help and error messages will be formatted to a line length of *width*.

`–a` Pathname must be an absolute path.

`–l` Pathname must be a relative path.

`–b` Pathname must be a block special file.

`–c` Pathname must be a character special file.

`–g` Pathname must be a regular file.

`–y` Pathname must be a directory.

`–n` Pathname must not exist (must be new).

`–o` Pathname must exist (must be old).

`–z` Pathname must have a length greater than 0 bytes.

`–r` Pathname must be readable.

`–t` Pathname must be creatable (touchable). Pathname will be created if it does not already exist.

`–w` Pathname must be writable.

`-x` Pathname must be executable.

`-d` Defines the default value as *default*. The default is not validated and so does not have to meet any criteria.

`-h` Defines the help messages as *help*.

`-e` Defines the error message as *error*.

`-p` Defines the prompt message as *prompt*.

`-k` Specifies that process ID *pid* is to be sent a signal if the user chooses to abort.

`-s` Specifies that the process ID *pid* defined with the `-k` option is to be sent signal `signal` when quit is chosen. If no signal is specified, `SIGTERM` is used.

input
Input to be verified against validation options.

EXIT CODES

`0` =Successful execution

`1` =EOF on input

`2` =Usage error

`3` =User termination (quit)

`4` =Mutually exclusive options

NOTES

The text of the default messages for `ckpath` depends upon the criteria options that have been used. A sample default prompt for `ckpath` (using the `-a` option) is:

```
Enter a pathname [?,q]:
```

A sample default error message (using the `-a` option) is:

```
ERROR - Invalid pathname entered. A pathname is a filename, optionally preceded
by parent directories.
```

An example default help message is:

```
A pathname is a filename, optionally preceded by parent directories.
The pathname you enter:
—— must contain 1 to {NAME_MAX} characters
—— must not contain a spaces or special characters
```

`NAME_MAX` is a system variable that is defined in `limits.h`.

When the quit option is chosen (and allowed), `q` is returned along with the return code `3`. The `valpath` module will not produce any output. It returns zero for success and non-zero for failure.

System Administrator's Reference Manual

ckrange(1M)

NAME

`ckrange, errange, helprange, valrange -` prompt forand validate an integer

SYNOPSIS

`/usr/bin/ckrange` [`–Q`] [`–W` *width*] [`–l` *lower*] [`–u` *upper*] [`–b` *base*] [`–d` *default*] [`–h` *help*] [`–e` *error*] [`–p` *prompt*] [`–k` *pid* [`–s` *signal*]]

`/usr/sadm/bin/errange` [`–W`] [`–l` *lower*] [`–u` *upper*] [`–e` *error*]

`/usr/sadm/bin/helprange` [`–W`] [`–l` *lower*] [`–u` *upper*] [`–h` *help*]

`/usr/sadm/bin/valrange` [`–l` *lower*] [`–u` *upper*] [`–b` *base*] *input*

DESCRIPTION

`ckrange` prompts a user and validates the response. It defines, among other things, a prompt message whose response should be an integer in the range specified, text for help and error messages, and a default value (which will be returned if the user responds with a carriage return).

This command also defines a range for valid input. If either the lower or upper limit is left undefined, then the range is bounded on only one end.

All messages are limited in length to 70 characters and are formatted automatically. Any white space used in the definition (including newline) is stripped. The `–W` option cancels the automatic formatting. When a tilde is placed at the beginning or end of a message definition, the default text will be inserted at that point, allowing both custom text and the default text to be displayed.

If the prompt, help or error message is not defined, the default message (as defined under **NOTES**) will be displayed.

Three visual tool modules are linked to the `ckrange` command. They are `errange` (which formats and displays an error message), `helprange` (which formats and displays a help message), and `valrange` (which validates a response). These modules should be used in conjunction with FACE objects. In this instance, the FACE object defines the prompt.

OPTIONS

The options and arguments for this command are:

`–Q` Specifies that quit will not be allowed as a valid response.

`–W` Specifies that prompt, help and error messages will be formatted to a line length of *width*.

`–l` Defines the lower limit of the range as *lower*. Default is the machine's largest negative integer or long.

`–u` Defines the upper limit of the range as *upper*. Default is the machine's largest positive integer or long.

`–b` Defines the base for input. Must be 2 to 36, default is 10.

`–d` Defines the default value as *default*. The default is not validated and so does not have to meet any criteria.

`–h` Defines the help messages as *help*.

`–e` Defines the error message as *error*.

`–p` Defines the prompt message as *prompt*.

`–k` Specifies that process ID *pid* is to be sent a signal if the user chooses to abort.

`–s` Specifies that the process ID *pid* defined with the `–k` option is to be sent signal `signal` when quit is chosen. If no signal is specified, `SIGTERM` is used.

input
Input to be verified against upper and lower limits and base.

EXIT CODES

`0` =Successful execution

`1` =EOF on input

`2` =Usage error

`3` =User termination (quit)

NOTES

The default base 10 prompt for `ckrange` is:

`Enter an integer between` *lower_bound* and *upper_bound* `[?,q]:`

The default base 10 error message is:

`ERROR - Please enter an integer between` *lower_bound* and *upper_bound*.

The default base 10 help message is:

`Please enter an integer between` *lower_bound* and *upper_bound*.

The messages are changed from "integer" to "base *base* integer" if the base is set to a number other than 10.

When the quit option is chosen (and allowed), `q` is returned along with the return code `3`. The `valrange` module will not produce any output. It returns zero for success and non-zero for failure.

System Administrator's Reference Manual

ckstr(1M)

NAME

`ckstr, errstr, helpstr, valstr -` display a prompt;verify and return a string answer

SYNOPSIS

`/usr/bin/ckstr` [`–Q`] [`–W` *width*] [[`–r` *regexp*] [...]] [`–l` *length*] [`–d` *default*] [`–h` *help*] [`–e` *error*] [`–p` *prompt*] [`–k` *pid* [`–s` *signal*]]

`/usr/sadm/bin/errstr` [`–W`] [`–e` *error*]

`/usr/sadm/bin/helpstr` [`–W`] [`–h` *help*]

`/usr/sadm/bin/valstr` *input*

DESCRIPTION

`ckstr` prompts a user and validates the response. It defines, among other things, a prompt message whose response should be a string, text for help and error messages, and a default value (which will be returned if the user responds with a carriage return).

The answer returned from this command must match the defined regular expression and be no longer than the length specified. If no regular expression is given, valid input must be a string with a length less than or equal to the length defined with no internal, leading or trailing white space. If no length is defined, the length is not checked. Either a regular expression or a length must be given with the command.

All messages are limited in length to 70 characters and are formatted automatically. Any white space used in the definition (including newline) is stripped. The `–W` option cancels the automatic formatting. When a tilde is placed at the beginning or end of a message definition, the default text will be inserted at that point, allowing both custom text and the default text to be displayed.

If the prompt, help or error message is not defined, the default message (as defined under **NOTES**) will be displayed.

Three visual tool modules are linked to the `ckstr` command. They are `errstr` (which formats and displays an error message), `helpstr` (which formats and displays a help message), and `valstr` (which validates a response). These modules should be used in conjunction with FACE objects. In this instance, the FACE object defines the prompt.

OPTIONS

The options and arguments for this command are:

`–Q` Specifies that quit will not be allowed as a valid response.

`–W` Specifies that prompt, help and error messages will be formatted to a line length of *width*.

`–r` Specifies a regular expression, `regexp`, against which the input should be validated. May include white space. If multiple expressions are defined, the answer must match only one of them.

`–l` Specifies the maximum length of the input.

`–d` Defines the default value as *default*. The default is not validated and so does not have to meet any criteria.

`–h` Defines the help messages as *help*.

`–e` Defines the error message as *error*.

`–p` Defines the prompt message as *prompt*.

`–k` Specifies that process ID *pid* is to be sent a signal if the user chooses to abort.

`–s` Specifies that the process ID *pid* defined with the `–k` option is to be sent signal `signal` when quit is chosen. If no signal is specified, `SIGTERM` is used.

input

Input to be verified against format length and/or regular expression criteria.

EXIT CODES

`0` =Successful execution

`1` =EOF on input

`2` =Usage error

`3` =User termination (quit)

NOTES

The default prompt for `ckstr` is:

```
Enter an appropriate value [?,q]:
```

The default error message is dependent upon the type of validation involved. The user will be told either that the length or the pattern matching failed.

The default help message is also dependent upon the type of validation involved. If a regular expression has been defined, the message is:

`Please enter a string which matches the following pattern:`*regexp*

Other messages define the length requirement and the definition of a string.

When the quit option is chosen (and allowed), `q` is returned along with the return code `3`. The `valstr` module will not produce any output. It returns zero for success and non-zero for failure.

System Administrator's Reference Manual

cktime(1M)

NAME

`cktime, errtime, helptime, valtime -` display aprompt; verify and return a time of day

SYNOPSIS

`/usr/bin/cktime` [`-Q`] [`-W` *width*] [`-f` *format*] [`-d` *default*] [`-h` *help*] [`-e` *error*] [`-p` *prompt*] [`-k` *pid* [`-s` *signal*]]

`/usr/sadm/bin/errtime` [`-W`] [`-e` *error*] [`-f` *format*]

`/usr/sadm/bin/helptime` [`-W`] [`-h` *help*] [`-f` *format*]

`/usr/sadm/bin/valtime` [`-f` *format*] *input*

DESCRIPTION

`cktime` prompts a user and validates the response. It defines, among other things, a prompt message whose response should be a time, text for help and error messages, and a default value (which will be returned if the user responds with a carriage return). The user response must match the defined format for the time of day.

All messages are limited in length to 70 characters and are formatted automatically. Any white space used in the definition (including newline) is stripped. The `-W` option cancels the automatic formatting. When a tilde is placed at the beginning or end of a message definition, the default text will be inserted at that point, allowing both custom text and the default text to be displayed.

If the prompt, help or error message is not defined, the default message (as defined under **NOTES**) will be displayed.

Three visual tool modules are linked to the `cktime` command. They are `errtime` (which formats and displays an error message), `helptime` (which formats and displays a help message), and `valtime` (which validates a response). These modules should be used in conjunction with FML objects. In this instance, the FML object defines the prompt. When *format* is defined in the `errtime` and `helptime` modules, the messages will describe the expected format.

OPTIONS

The options and arguments for this command are:

`-Q` Specifies that quit will not be allowed as a valid response.

`-W` Specifies that prompt, help and error messages will be formatted to a line length of *width*.

`-f` Specifies the format against which the input will be verified. Possible formats and their definitions are:

- `%H` =hour (00 - 23)
- `%I` =hour (00 - 12)
- `%M` =minute (00 - 59)
- `%p` =ante meridian or post meridian
- `%r` =time as `%I:%M:%S %p`
- `%R` =time as `%H:%M` (the default format)
- `%S` =seconds (00 - 59)
- `%T` =time as `%H:%M:%S`

`-d` Defines the default value as *default*. The default is not validated and so does not have to meet any criteria.

`-h` Defines the help messages as *help*.

`-e` Defines the error message as *error*.

`-p` Defines the prompt message as *prompt*.

`-k` Specifies that process ID *pid* is to be sent a signal if the user chooses to abort.

`-s` Specifies that the process ID *pid* defined with the `-k` option is to be sent signal `signal` when quit is chosen. If no signal is specified, `SIGTERM` is used.

input
Input to be verified against format criteria.

EXIT CODES

`0` =Successful execution

`1` =EOF on input

`2` =Usage error

`3` =User termination (quit)

`4` =Garbled format argument

NOTES

The default prompt for `cktime` is:

```
Enter the time of day [?,q]:
```

The default error message is:

```
ERROR - Please enter the time of day, using the following format:format
```

The default help message is:

```
Please enter the time of day, using the following format:format
```

When the quit option is chosen (and allowed), `q` is returned along with the return code `3`. The `valtime` module will not produce any output. It returns zero for success and non-zero for failure.

System Administrator's Reference Manual

ckuid(1M)

NAME

`ckuid, erruid, helpuid, valuid -` prompt for andvalidate a user ID

SYNOPSIS

`/usr/bin/ckuid [ –Q ] [ –W` *width* `] [ –m ] [ –d` *default* `] [ –h` *help* `] [ –e` *error* `] [ –p` *prompt* `] [ –k` *pid* `[ –s` *signal* `]]`

`/usr/sadm/bin/erruid [ –W ] [ –e` *error* `]`

`/usr/sadm/bin/helpuid [ –W ] [ –m ] [ –h` *help* `]`

`/usr/sadm/bin/valuid` *input*

DESCRIPTION

`ckuid` prompts a user and validates the response. It defines, among other things, a prompt message whose response should be an existing user ID, text for help and error messages, and a default value (which will be returned if the user responds with a carriage return).

All messages are limited in length to 70 characters and are formatted automatically. Any white space used in the definition (including newline) is stripped. The `–W` option cancels the automatic formatting. When a tilde is placed at the beginning or end of a message definition, the default text will be inserted at that point, allowing both custom text and the default text to be displayed.

If the prompt, help or error message is not defined, the default message (as defined under **NOTES**) will be displayed.

Three visual tool modules are linked to the `ckuid` command. They are `erruid` (which formats and displays an error message), `helpuid` (which formats and displays a help message), and `valuid` (which validates a response). These modules should be used in conjunction with FML objects. In this instance, the FML object defines the prompt.

OPTIONS

The options and arguments for this command are:

`–Q` Specifies that quit will not be allowed as a valid response.

`–W` Specifies that prompt, help and error messages will be formatted to a line length of *width*.

`–m` Displays a list of all logins when help is requested or when the user makes an error.

`–d` Defines the default value as *default*. The default is not validated and so does not have to meet any criteria.

`–h` Defines the help messages as *help*.

`–e` Defines the error message as *error*.

`–p` Defines the prompt message as *prompt*.

`–k` Specifies that process ID *pid* is to be sent a signal if the user chooses to abort.

`–s` Specifies that the process ID *pid* defined with the `–k` option is to be sent signal `signal` when quit is chosen. If no signal is specified, `SIGTERM` is used.

input

Input to be verified against `/etc/passwd`.

EXIT CODES

`0` =Successful execution

`1` =EOF on input

`2` =Usage error

`3` =User termination (quit)

NOTES

The default prompt for `ckuid` is:

```
Enter the login name of an existing user [?,q]:
```

The default error message is:

```
ERROR - Please enter the login name of an existing user.
Select the help option (?) for a list of valid login names.
```

(Last line appears only if the `–m` option of `ckuid` is used.)

The default help message is:

```
Please enter the login name of an existing user.
```

(If the `–m` option of `ckuid` is used, a list of valid groups is also displayed.)

When the quit option is chosen (and allowed), `q` is returned along with the return code `3`. The `valuid` module will not produce any output. It returns zero for success and non-zero for failure.

System Administrator's Reference Manual

ckyorn(1M)

NAME

`ckyorn, erryorn, helpyorn, valyorn -` prompt for andvalidate yes/no

SYNOPSIS

`/usr/bin/ckyorn [ –Q ] [ –W` *width* `] [ –d` *default* `] [ –h` *help* `] [ –e` *error* `] [ –p` *prompt* `] [ –k` *pid* `[ –s` *signal* `]]`

`/usr/sadm/bin/erryorn [ –W ] [ –e` *error* `]`

`/usr/sadm/bin/helpyorn [ –W ] [ –h` *help* `]`

`/usr/sadm/bin/valyorn` *input*

DESCRIPTION

`ckyorn` prompts a user and validates the response. It defines, among other things, a prompt message for a yes or no answer, text for help and error messages, and a default value (which will be returned if the user responds with a carriage return).

All messages are limited in length to 70 characters and are formatted automatically. Any white space used in the definition (including newline) is stripped. The `–W` option cancels the automatic formatting. When a tilde is placed at the beginning or end of a message definition, the default text will be inserted at that point, allowing both custom text and the default text to be displayed.

If the prompt, help or error message is not defined, the default message (as defined under **NOTES**) will be displayed.

Three visual tool modules are linked to the `ckyorn` command. They are `erryorn` (which formats and displays an error message), `helpyorn` (which formats and displays a help message), and `valyorn` (which validates a response). These modules should be used in conjunction with FACE objects. In this instance, the FACE object defines the prompt.

OPTIONS

The options and arguments for this command are:

`–Q` Specifies that quit will not be allowed as a valid response.

`–W` Specifies that prompt, help and error messages will be formatted to a line length of *width*.

`–d` Defines the default value as *default*. The default is not validated and so does not have to meet any criteria.

`–h` Defines the help messages as *help*.

`–e` Defines the error message as *error*.

`–p` Defines the prompt message as *prompt*.

`–k` Specifies that process ID *pid* is to be sent a signal if the user chooses to abort.

`–s` Specifies that the process ID *pid* defined with the `–k` option is to be sent signal `signal` when quit is chosen. If no signal is specified, `SIGTERM` is used.

input

Input to be verified as `y`, `yes`, `Y`, `Yes`, `YES` or `n`, `no`, `N`, `No`, `NO`.

EXIT CODES

`0` =Successful execution

`1` =EOF on input

`2` =Usage error

`3` =User termination (quit)

NOTES

The default prompt for `ckyorn` is:

```
Yes or No [y,n,?,q]:
```

The default error message is:

```
ERROR - Please enter yes or no.
```

The default help message is:

```
To respond in the affirmative, enter y, yes, Y, or YES.
To respond in the negative, enter n, no, N, or NO.
```

When the quit option is chosen (and allowed), `q` is returned along with the return code `3`. The `valyorn` module will not produce any output. It returns zero for success and non-zero for failure.

System Administrator's Reference Manual

colltbl(1M)

NAME

`colltbl` - create collation database

SYNOPSIS

`colltbl` [*file*|–]

DESCRIPTION

The `colltbl` command takes as input a specification file, *file*, that describes the collating sequence for a particular language and creates a database that can be read by `strxfrm(3C)` and `strcoll(3C)`. `strxfrm(3C)` transforms its second argument and places the result in its first argument. The transformed string is such that it can be correctly ordered with other transformed strings by using `strcmp(3C)`, `strncmp(3C)` or `memcmp(3C)`. `strcoll(3C)` transforms its arguments and does a comparison.

If no input file is supplied, standard input is read.

The output file produced contains the database with collating sequence information in a form usable by system commands and routines. The name of this output file is the value you assign to the keyword `codeset` read in from *file*. Before this file can be used, it must be installed in the `/usr/lib/locale/`*locale* directory with the name `LC_COLLATE` by someone who is superuser or a member of group `bin`. *locale* corresponds to the language area whose collation sequence is described in *file*. This file must be readable by user, group, and other; no other permissions should be set. To use the collating sequence information in this file, set the `LC_COLLATE` environment variable appropriately [see `environ(5)` or `setlocale(3C)`].

The `colltbl` command can support languages whose collating sequence can be completely described by the following cases:

- Ordering of single characters within the codeset. For example, in Swedish, `V` is sorted after `U`, before `X` and with `W` (`V` and `W` are considered identical as far as sorting is concerned).
- Ordering of "double characters" in the collation sequence. For example, in Spanish, `ch` and `ll` are collated after `c` and `l`, respectively.
- Ordering of a single character as if it consists of two characters. For example, in German, the "sharp s", `ß`, is sorted as `ss`. This is a special instance of the next case below.
- Substitution of one character string with another character string. In the example above, the string `ß` is replaced with `ss` during sorting.
- Ignoring certain characters in the codeset during collation. For example, if the hyphen "–" were ignored during collation, then the strings `re-locate` and `relocate` would be equal.
- Secondary ordering between characters. In the case where two characters are sorted together in the collation sequence, (i.e. they have the same "primary" ordering), there is sometimes a secondary ordering that is used if two strings are identical except for characters that have the same primary ordering. For example, in French, the letters `e` and `è` have the same primary ordering but `e` comes before `è` in the secondary ordering. Thus the word `des` would be ordered before `dès`.

The specification file consists of three types of statements:

1. `codeset` *filename*

   filename is the name of the output file to be created by `colltbl`.

2. `order is` *order_list*

   order_list is a list of symbols, separated by semicolons, that defines the collating sequence. The special symbol ". . ." specifies symbols that are lexically sequential in a short-hand form. For example,

   `order is a;b;c;d; . . . ;x;y;z`

   would specify the list of lowercase letters. Of course, this could be further compressed to just `a; . . . ;z`.

A symbol can be up to two bytes in length and can be represented in any one of the following ways:

- the symbol itself (e.g. `a` for the lowercase letter `a`),
- in octal representation (e.g. `\141` or `0141` for the letter `a`), or
- in hexadecimal representation (e.g. `\x61` or `0x61` for the letter `a`).

Any combination of these may be used as well.

The backslash character (`\`) is used for continuation. No characters are permitted after the backslash character.

Symbols enclosed in parentheses are assigned the same primary ordering but different secondary ordering. Symbols enclosed in curly brackets are assigned only the same primary ordering. For example,

```
order is a;b;c;ch;d;(e;è);f; . . . ;z;\
{1; . . . ;9};A; . . . ;Z
```

In the above example, `e` and `è` are assigned the same primary ordering and different secondary ordering, digits 1 through 9 are assigned the same primary ordering and no secondary ordering. Only primary ordering is assigned to the remaining symbols. Notice how double letters can be specified in the collating sequence (letter `ch` comes between `c` and `d`).

If a character is not included in the `order is` statement it is excluded from the ordering and will be ignored during sorting.

3. `substitute` *string* `with` *repl*

   The `substitute` statement substitutes the string *string* with the string *repl*. This can be used, for example, to provide rules to sort the abbreviated month names numerically:

   ```
   substitute "Jan" with "01"
   substitute "Feb" with "02"
   . . .
   substitute "Dec" with "12"
   ```

   A simpler use of the `substitute` statement that was mentioned above was to substitute a single character with two characters, as with the substitution of `ß` with `ss` in German.

The `substitute` statement is optional. The `order is` and `codeset` statements must appear in the specification file.

Any lines in the specification file with a `#` in the first column are treated as comments and are ignored. Blank lines are also ignored.

EXAMPLES

The following example shows the collation specification required to support a hypothetical telephone book sorting sequence.

The sorting sequence is defined by the following rules:

a) Uppercase and lowercase letters must be sorted together, but uppercase letters have precedence over lowercase letters.

b) All special characters and punctuation should be ignored.

c) Digits must be sorted as their alphabetic counterparts (e.g. `0` as `zero`, `1` as `one`).

d) The `Ch`, `ch`, `CH` combinations must be collated between `C` and `D`.

e) `V` and `W`, `v` and `w` must be collated together.

The input specification file to `colltbl` will contain:

```
codeset telephone
order is A;a;B;b;C;c;CH;Ch;ch;D;d;E;e;F;f;\
G;g;H;h:I;i;J;j;K;k;L;l;M;m;N;n;O;o;P;p;\
Q;q;R;r;S;s;T;t;U;u;{V;W};{v;w};X;x;Y;y;Z;z
substitute "0" with "zero"
```

```
substitute "1" with "one"
substitute "2" with "two"
substitute "3" with "three"
substitute "4" with "four"
substitute "5" with "five"
substitute "6" with "six"
substitute "7" with "seven"
substitute "8" with "eight"
substitute "9" with "nine"
```

LOCALE

The LC_MESSAGES environment variable governs the language in which message texts are displayed.

The LC_CTYPE environment variable governs character classes, character conversion (shifting), and the behavior of character classes in regular expressions.

If LC_MESSAGES or LC_CTYPE is undefined or is defined as the null string, it defaults to the value of LANG. If LANG is likewise undefined or null, the system acts as if it were not internationalized.

If any of the locale variables has an invalid value, the system acts as if none of the variables was set.

The LC_ALL environment variable governs the entire locale. LC_ALL takes precedence over all the other environment variables that affect internationalization.

FILES

/lib/locale/*locale*/LC_COLLATE
: LC_COLLATE database for *locale*

/usr/lib/locale/C/colltbl_C
: input file used to construct LC_COLLATE in the default locale

SEE ALSO

memory(3C), setlocale(3C), strcoll(3C), string(3C), strxfrm(3C), environ(5).

System Administrator's Reference Manual

comsat(1M)

NAME

`comsat, in.comsat -` biff server

SYNOPSIS

`in.comsat`

DESCRIPTION

`comsat` is the server process which listens for reports of incoming mail and notifies users who have requested to be told when mail arrives. It is invoked as needed by `inetd`(1M) , and times out if inactive for a few minutes.

`comsat` listens on a datagram port associated with the `biff` service specification [see `services`(4)] for one line messages of the form

user@mailbox–offset

If the *user* specified is logged in to the system and the associated terminal has the owner execute bit turned on (by a `biff y`), the *offset* is used as a seek offset into the appropriate mailbox file and the first 7 lines or 560 characters of the message are printed on the user's terminal. Lines which appear to be part of the message header other than the `From`, `To`, `Date`, or `Subject` lines are not printed when displaying the message.

NOTES

The message header filtering is prone to error.

FILES

`/var/utmp`
: who's logged on and on what terminals

SEE ALSO

`inetd`(1M) , `services`(4) .

System Administrator's Reference Manual

contmerg(1M)

NAME

`contmerg` - merge contents files

SYNOPSIS

`contmerg` *contents_old*

DESCRIPTION

The `contmerg` command is used to merge a *contents_old* file with the standard `contents` file. If the object names in both `contents` files are the same, the entry in the standard `contents` file is regarded as being more recent. The package and class affiliations from the *contents_old* file are always preserved however.

The *contents_old* file must be specified as a full path name.

The command is used to speed up the operating system update: Before the update, the `contents` file is renamed *contents_old* and the new `contents` file that results from the update is merged with *contents_old* using the `contmerg` command.

System Administrator's Reference Manual

crash(1M)

NAME

`crash` - examine system images

SYNOPSIS

`/usr/sbin/crash` [`–d` *dumpfile*] [`–n` *namelist*] [`–w` *outputfile*]

DESCRIPTION

The `crash` command is used to examine the system memory image of a running or a crashed system by formatting and printing control structures, tables, and other information. Command line arguments for `crash` are *dumpfile*, *namelist*, and *outputfile*.

dumpfile is the file containing the system memory image. The default *dumpfile* is `/dev/mem`.

The text file *namelist* contains the symbol table information needed for symbolic access to the system memory image to be examined. The default *namelist* is `/stand/unix`. If a system image from another machine is to be examined, the corresponding text file must be copied from that machine.

When the `crash` command is invoked, a session is initiated. The output from a `crash` session is directed to *outputfile*. The default *outputfile* is the standard output.

Input during a `crash` session is of the form:

function [*argument* ...]

where *function* is one of the `crash` functions described in the **FUNCTIONS** subsection of this manual page, and *arguments* are qualifying data that indicate which items of the system image are to be printed.

The default for process-related items is the current process for a running system or the process that was running at the time of the crash for a crashed system. If the contents of a table are being dumped, the default is all active table entries.

OPTIONS

The following function options are available to `crash` functions wherever they are semantically valid.

`–e` Shows every entry in a table.

`–f` Shows the full structure.

`–p` Interprets all address arguments in the command line as physical addresses. If they are not physical addresses, results are inconsistent.

`–s` *process*
Specifies a process slot other than the default.

`–w` *file*
Redirects the output of a function to *file*.

The functions `mode`, `defproc`, and `redirect` correspond to the function options `–p`, `–s`, and `–w`. The `mode` function may be used to set the address translation mode to physical or virtual for all subsequently entered functions; `defproc` sets the value of the process slot argument for subsequent functions; and `redirect` redirects all subsequent output.

Output from `crash` functions may be piped to another program in the following way:

function [*argument* ...] ! *shell_command*

For example,

```
mount ! grep rw
```

writes all mount table entries with an `rw` flag to standard output. The redirection option (`–w`) cannot be used with this feature.

Depending on the context of the function, numeric arguments are assumed to be in a specific radix. Counts

are assumed to be decimal. Addresses are always hexadecimal. Table address arguments larger than the size of the function table are interpreted as hexadecimal addresses; those smaller are assumed to be decimal slots in the table. Default bases on all arguments may be overridden. The C conventions for designating the bases of numbers are recognized. A number that is usually interpreted as decimal is interpreted as hexadecimal if it is preceded by `0x` and as octal if it is preceded by `0`. Decimal override is designated by `0d`, and binary by `0b`.

Aliases for functions may be any uniquely identifiable initial substring of the function name. Traditional aliases of one letter, such as `p` for `proc`, remain valid.

Many functions accept different forms of entry for the same argument. Requests for table information accept a table entry number, a physical address, a virtual address, a symbol, a range, or an expression. A range of slot numbers may be specified in the form *a*–*b* where *a* and *b* are decimal numbers. An expression consists of two operands and an operator. An operand may be an address, a symbol, or a number; the operator may be +, –, *, /, &, or |. An operand that is a number should be preceded by a radix prefix if it is not a decimal number (`0` for octal, `0x` for hexadecimal, `0b` for binary). The expression must be enclosed in parentheses. Other functions accept any of these argument forms that are meaningful.

Two abbreviated arguments to `crash` functions are used throughout. Both accept data entered in several forms. They may be expanded into the following:

table_entry = *table_entry* | *address* | *symbol* | *range* | *expression*
start_addr = *address* | *symbol* | *expression*

FUNCTIONS

`?` [–w *file*]
List available functions.

`!`*command*
Escape to the shell and execute *command*.

`as` [–e] [–f] [–w *file*] [*proc* ...]
Print information on process segments.

`async` [–f] [–w *file*]
Output the list of data structures for asynchronous input/output requests.

`autoconf` [–f] [–w *file*]
Gives information about the hardware components of a system. This function is only supported during testing of system storage dumps.

`base` [–w *file*] *number* ...
Print *number* in binary, octal, decimal, and hexadecimal. A number in a radix other than decimal should be preceded by a prefix that indicates its radix as follows: `0x`, hexadecimal; `0`, octal; and `0b`, binary.

`buffer` [–w *file*] [–*format*] *bufferslot*
`buffer` [–w *file*] [–format] [–p] *start_addr*
Alias: `b`.

Print the contents of a buffer in the designated format. The following format designations are recognized: –b, byte; –c, character; –d, decimal; –x, hexadecimal; –o, octal; and, –i, inode. If no format is given, the previous format is used. The default format at the beginning of a `crash` session is hexadecimal.

`bufhdr` [–f] [–w *file*] [[–p] *table_entry* ...]
Alias: `buf`.

Print system buffer headers. The –f option produces different output depending on whether the buffer is local or remote (contains RFS data).

`bsdso` [–w *file*] [–p] *address* ...
Output the `struct socket` data structure for the specified addresses.

`bt` [–w *file*] [–f] [–p *table_entry*] *stack_address*
Show a stack log starting with the *stack_address*.

`callout` [–w *file*]
Alias: `c`.
Print the callout table.

`class` [–w *file*] [*table_entry* ...]
Print information about process scheduler classes.

`cpupriv` [–w *file*] [*cpu_number*]
Output the `struct percpu` data structure for the specified CPU.

`crawl` [–w *file*] [–s *start_address*] [*table_entry*]
`crawl` [–w *file*] [–s *start_address*] [#*procid*]
`crawl` [–w *file*] [–s *start_address*] [*proc_addr*]
Search the stack log for possible stack entries and disassemble the area of 4 commands around the call of the next routine. You can also specify the start address from which the stack log is to be searched and specify the process.

`dbfree` [–w *file*] [*class* ...]
Print free streams data block headers. If a class is entered, only data block headers for the class specified is printed.

`dblock` [–e] [–w *file*] [–c *class* ...]
`dblock` [–e] [–w *file*] [[–p] *table_entry* ...]
Print allocated streams data block headers. If the class option (`–c`) is used, only data block headers for the class specified is printed.

`defproc` [–w *file*] [–c]
`defproc` [–w *file*] [*slot*]
Set the value of the process slot argument. The process slot argument may be set to the current slot number (`–c`) or the slot number may be specified. If no argument is entered, the value of the previously set slot number is printed. At the start of a `crash` session, the process slot is set to the current process.

`devip` [–w *file*] [–p] [*devip_providers_addr* ...]
Show the specified `devip_providers` structure or, if no addresses are specified, all `devip_providers` structures used. The command behaves like `prov` on the fallback kernel.

`dis` [–w *file*] [–a] *start_addr* [*count*]
`dis` [–w *file*] [–a] –c [*count*]
Disassemble *count* instructions starting at *start_addr*. The default count is 1. The absolute option (`–a`) specifies a non-symbolic disassembly. The `–c` option can be used in place of *start_addr* to continue disassembly at the address at which a previous disassembly ended.

`dispq` [–w *file*] [*table_entry* ...]
Print the dispatcher (scheduler) queues.

`ds` [–w *file*] *virtual_address* ...
Print the data symbol whose address is closest to, but not greater than, the address entered.

`ep` [–c`Streams`] [–w*filename*] [–a*age*]
Evaluation of the profile of the `ep`(7) event profiler.

`–c` Specify the *CLIENT* (only `Streams` at present).

`–w` Redirect output to *filename*.

`–a` Only events older than *age* are output. *age* can be specified in

n[`h`] hours

n[`m`] minutes

n[`s`] seconds

Example: *age* = `1h` or `30m` or `10s`

If *age* is specified without a unit, the value is interpreted as a tic (1/`HZ` sec.).

A heading is output first specifying the *CLIENT* and the tic value since the system boot. The `crash` version and the mode of the profile are then output.

The stored information is then evaluated. The event profiler first prints the *OBJECT* key in every line. This makes it easy to process the output at a later point with `grep`, `sed` or `awk`.

The first event output relates to creating the *OBJECT* [`ep_create()`]. All other events were stored by `ep_push()` calls.

The second column contains the tics since the system boot. The third column contains the trace IDs and the names of the trace points in case `crash` knows these.

The next columns either contain the stack backtrace of the caller or the evaluation of trace-specific data.

`ep_callers` [`–cStreams`] `–s`*callers*

Determine the number of callers to be stored in the stack backtrace of the caller.

`–c` Specify the *CLIENT* (only `Streams` at present).

`–s` Number of callers [0,...,10]

The event profiler normally stores five callers. There may be cases, however, where it suffices to store fewer callers. Since the definition of callers is very time-consuming for MIPS, the number of callers should be kept to a minimum so as not to interrupt operation too severely.

`ep_cleanup` [`–cStreams`]

Free all memory for the *CLIENT* again.

`–c` Specify the *CLIENT* (only `Streams` at present).

`ep_init –cStreams –m{PROFILING | MFREE}` [`–t`*mask*] [`–p`] [`–o`*objects*] [`–e`*events*]

Initialize event profiler `ep(7)` and reserve memory for the profile.

`–c` Specify the *CLIENT* (only `Streams` at present).

`–m` Mode: Either `PROFILING` (history of all *OBJECTS*) created or `MFREE` (multiple releases discovered).

`–t` Bitmask in which the bits are set for all active trace points.

`–p` A PANIC should be triggered when the configured number of *OBJECTS* has been reached.

`–o` Max. number of *OBJECTS*, for which information can be stored at the same time [4000,...,40000].

`–e` Max. number of events that can be stored simultaneously [10000,...,100000].

The `–p` option can be very helpful in locating the memory leak. If you wait until the system has freed the memory, it is generally no longer possible to generate a system dump. However, if the event profiler triggers a PANIC as soon as the number of *OBJECTS* has been reached, there are generally sufficient events stored in the event profiler to find the memory leak. The dump can then be evaluated subsequently using `crash`.

If no optional parameters were specified, `ep_init` uses appropriate default values which depend on the *CLIENT* and the mode. The profile then needs approx. 4 MB main memory.

`ep_start` [`–cStreams`]

Start the `ep(7)` event profiler for the *CLIENT*: The storage of events begins.

`–c` Specify the *CLIENT* (only `Streams` at present).

`ep_stop` [`–cStreams`]

Halt the event profiler `ep(7)` for the *CLIENT* again: The storage of creation and other events is halted. Processing of the release events continues as otherwise alleged memory leaks would be displayed.

The profile can then be evaluated with the `crash` command `ep`, without fear of the data being evaluated being modified permanently in the process.

Note:

A streams client cannot be halted in `MFREE` mode using `ep_stop`. Use the `ep_cleanup` command to

abort the Event Profiler and release the memory used for logging.

`-c` Specify the *CLIENT* (only `Streams` at present).

`evactive` [`-w` *file*] [`-f`] [*event_name*]
Output all event queues. The output can also be filtered according to the *event_name*.

`evmm` [`-w` *file*]
Output the administration structures of the event memory management.

`file` [`-e`] [`-w` *file*] [[`-p`] *table_entry* ...]
Alias: `f`.

Print the file table.

`fileof` [`-w` *file*] *inpcb_addr*
Search for processes/file descriptors that refer to the `struct msgb` *inpcb_addr*.

`findaddr` [`-w` *file*] *table slot*
Print the address of *slot* in *table*. Only tables available to the `size` function are available to `findaddr`.

`findslot` [`-w` *file*] *virtual_address* ...
Print the table, entry slot number, and offset for the address entered. Only tables available to the `size` function are available to `findslot`.

`fs` [`-w` *file*] [[`-p`] *table_entry* ...]
Print the file system information table.

`gdp` [`-e`] [`-f`] [`-w` *file*] [[`-p`] *table_entry* ...]
Print the gift descriptor protocol table. A "gift" denotes a reference to a remote receive file descriptor.

`help` [`-w` *file*] *function* ...
Print a description of the named function, including syntax and aliases.

`hrt` [`-w` *file*]
Output all high-resolution timers.

`icmpstat` [`-w` *file*] [[`-p`] *icmpstat_addr* ...]
Output the ICMP (Interface Control Message Protocol) statistic.

`ifnet` [`-w` *file*] [`-l`] [`-p`] *ifnet_addr* ...
Output the `struct ifnet` (network interface statistic) data structure for the specified *ifnet_addr* addresses.

`ifstats` [`-w` *file*] [`-l`] [`-p`] *ifstats_addr* ...
Output the `struct ifstats` (interfaces statistic) data structure for the specified *ifstats_addr* addresses.

`ikdb` Switch to `ikdb` emulation mode. All functions called subsequently correspond to those of the `ikdb` diagnostics program which is contained in the system kernel. Deviations from this are described here.

`inode` [`-e`] [`-f`] [`-w` *file*] [[`-p`] *table_entry* ...]
Alias: `i`.

Print the inode table, including file system switch information.

`inpcb` [`-w` *file*] [`-l`] [`-f`] [`-p`] *inpcb_addr* ...
Output the `struct inpcb` (Internet Protocol Control Block) data structure for the specified *inpcb_addr* addresses. The host entries for the entire list of these data structures are output if the `-l` option is specified. Detailed information is output with the list if the `-f` option is also specified.

`ipcs` [`-w` *file*] [`-f`] [`-m`] [`-s`] [`-q`]
Output information on active facilities for interprocess communication (IPC facilities). You can control the type of IPC facilities used to output information by specifying options: message queues (`-q`), semaphores (`-s`), or shared memory (`-m`).

`ipstat` [`-w` *file*] [[`-p`] *ipstat_addr* ...]
Output the Internet protocol status (`struct ipstat` data structure) for the specified *ipstat_addr*

addresses.

`kfp` [`–w` *file*] [`–s` *process*] [`–r`]
`kfp` [`–w` *file*] [`–s` *process*] [*value*]
Print the kernel frame pointer (kfp) for the start of a kernel stack trace. The kfp value can be set using the value argument or the reset option (`–r`), which sets the kfp through the NVRAM (non-volatile RAM). If no argument is entered, the current value of the kfp is printed.

`kmafreelists` [`–w` *file*] [`–f`]
Output the free lists for the kernel memory.

`kmastat` [`–w` *file*]
Print kernel memory allocator statistics.

`lck` [`–e`] [`–w` *file*] [[`–p`] *table_entry* ...]
Alias: `l`.

Print record locking information. If the `–e` option is used or table address arguments are given, the record lock list is printed. If no argument is entered, information on locks relative to inodes is printed.

`linkblk` [`–e`] [`–w` *file*] [[`–p`] *table_entry* ...]
Print the linkblk table.

`map` [`–w` *file*] *mapname* ...
Print the map structure of the given mapname.

`mb2info` [`–w` *file*] *mb2info_addr*
Output the `struct mb2_ctlr_info` MBII controller information data structure for the specified *mb2info_addr* address. This function is only supported for the RM600-xxx.

`mbfree` [`–w` *file*]
Print free streams message block headers.

`mblk` [`–w` *file*] [`–c`] [`–d`] [`–f`] [`–r`] [`–p`] *mblk_addr* ...
Output information on streams messages. All message blocks consisting of a single logical message are output using the `–c` option. Data belonging to the message block is output, in addition, using the `–d` option. Message blocks can be output in the sequence of forward chaining (`–f`) or reverse chaining (`–r`).

`mblock` [`–e`] [`–w` *file*] [[`–p`] *table_entry* ...]
Print allocated streams message block headers.

`mode` [`–w` *file*] [*mode*]
Set address translation of arguments to virtual (`v`) or physical (`p`) mode. If no mode argument is given, the current mode is printed. At the start of a `crash` session, the mode is virtual.

`mount` [`–e`] [`–w` *file*] [[`–p`] *table_entry* ...]
Alias: `m`, `vfs`.

Print information about mounted file systems.

`mplck` [`–w` *file*] [`–p`] *mplck_addr* ...
Output information on multiprocessor locks (`struct lck`).

`mplock` [`–w` *file*] [`–p`] *mplck_addr* ...
Output information on multiprocessor locks (`struct lck`).

`mplog` [`–w` *file*] [`–f`] [`–r`] [`–l` *mplck_addr*] [*count*]
Output logged information on multiprocessor locks. The output is unformatted if the `–r` option is specified. The `–l` option restricts output to the *mplck_addr* lock. The number of log entries to be output can be specified using *count*.

`mpsema` [`–w` *file*] [`–p`] *lock_addr* ...
Output information on the *lock_addr* semaphore.

`mpspin` [`–w` *file*] [`–p`] *lock_addr* ...

Output information on the *lock_addr* multiprocessor lock.

`mpstat` [`–w` *file*]
Output status information on multiprocessor locks.

`myq` [`–w` *file*] [`–p`] *queue_addr* ...
Output the administration structures for message queues (`struct queue` data structure).

`nm` [`–w` *file*] *symbol* ...
Print value and type for the given symbol.

`od` [`–p`] [`–w` *file*] [*–format*] [*–mode*] [`–s` *process*] *start_addr* [*count*]
Alias: `rd`.

Print *count* values starting at *start_addr* in one of the following formats: character (`–c`), decimal (`–d`), hexadecimal (`–x`), octal (`–o`), ASCII (`–a`), or hexadecimal character (`–h`), and one of the following modes: long (`–l`), int (`–i`), short (`–t`), or byte (`–b`). The default mode for character and ASCII formats is byte; the default mode for decimal, hexadecimal, and octal formats is long. The format `–h` prints both hexadecimal and character representations of the addresses dumped; no mode needs to be specified. When format or mode is omitted, the previous value is used. At the start of a `crash` session, the format is hexadecimal and the mode is long. If no count is entered, 1 is assumed.

`page` [`–e`] [`–w` *file*] [[`–p`] *table_entry* ...]
Print information about pages.

`pcb` [`–w` *file*] [`–u`] [*process*]
`pcb` [`–w` *file*] [`–k`] [*process*]
`pcb` [`–w` *file*] [[`–p`]`–i` *start_addr*]
Print the process control block. If no arguments are given, the active pcb for the current process is printed. The user option (`–u`) prints the user pcb and the kernel option (`–k`) prints the kernel pcb associated with the process. The interrupt option (`–i`) prints the interrupt pcb located at *start_addr*.

`plock` [`–t`]
`plock` [`–d`]
`plock` [`–u`]
This command can be used to lock the text segment (`–t`) or the data segment (`–d`) of the `crash` process in the memory. There is no option for locking both segments in the memory. This means that the `crash` process can no longer be swapped out. The `–u` option cancels existing locks.

`prnode` [`–e`] [`–w` *file*] [[`–p`] *table_entry* ...]
Print information about the private data of processes being traced.

`proc` [`–e`] [`–f`] [`–w` *file*] [[`–p`] [`–a`] *table_entry* ... `#`*procid* ...]
`proc` [`–f`] [`–w` *file*] [`–r`]
Alias: `p`.

Print the process table. Process table information may be specified in two ways. First, any mixture of table entries and process IDs may be entered. Each process ID must be preceded by a `#`. Alternatively, process table information for executable processes may be specified with the executable option (`–r`). Only those processes that have alarm times set are output if the `–a` option is specified.

`prov` [`–w` *file*] [[`–p`] *ip_provider_addr* ...]
Output all or specific (*ip_provider_addr*) `struct ip_provider` data structures. The command behaves like `devip` in the case of an enhanced kernel.

`provext` [`–w` *file*]
Output all `struct ip_provider_ext` data structures.

`ptbl` [`–w` *file*] [`–s` *process*] *section segment* [*count*]
`ptbl` [`–w` *file*] [`–s` *process*] [`–p`] *addr* [*count*]
Print information on page descriptor tables.

`pte` [`–w` *file*] *pte_addr* ...
Output the `struct pte` data structure for the specified *pte_addr* addresses.

`pty` [`–f`] [`–e`] [`–w` *file*] [`–s`] [`–h`] [`–l`] [[`–p`] *table_entry* ...]
Print the pseudo ttys presently configured. The `–l`, `–h` and `–h` options give information about the STREAMS modules `ldterm`, `ptem` and `pckt`, respectively.

`qrun` [`–w` *file*]
Print the list of scheduled streams queues.

`queue` [`–e`] [`–w` *file*] [[`–p`] *table_entry* ...]
Print streams queues.

`quit` Alias: `q`.
Terminate the `crash` session.

`rcvd` [`–e`] [`–f`] [`–w` *file*] [[`–p`] *table_entry* ...]
Print the receive descriptor table.

`rduser` [`–e`] [`–f`] [`–w` *file*] [[`–p`] *table_entry* ...]
Print the receive descriptor user table.

`redirect` [`–w` *file*] [`–c`]
`redirect` [`–w` *file*] [*newfile*]
Used with a file name, redirects output of a `crash` session to *newfile*. If no argument is given, the file name to which output is being redirected is printed. Alternatively, the close option (`–c`) closes the previously set file and redirects output to the standard output.

`resource` [`–e`] [`–w` *file*]
Print the advertise table.

`route` [`–w` *file*] [`–p`] [*route_addr* ...]
Output a list or specific (*route_addr*) data structures of type `struct route`.

`rtdptbl` [`–w` *file*] [*table_entry* ...]
Print the real-time scheduler parameter table [see `rt_dptbl(4)`].

`rtentry` [`–w` *file*] [`–p`] *rtentry_addr* ...
Output the `struct rtentry` data structure for the specified *rtentry_addr* addresses.

`rtproc` [`–w` *file*]
Print information about processes in the real-time scheduler class.

`search` [`–p`] [`–w` *file*] [`–m` *mask*] [`–s` *process*] *pattern start_addr length*
Print the words in memory that match *pattern*, beginning at the *start_addr* for *length* words. The mask is ANDed (&) with each memory word and the result compared against the pattern. The mask defaults to `0xffffffff`.

`size` [`–w` *file*] [`–x`] [*structure_name* ...]
Print the size of the designated structure. The (`–x`) option prints the size in hexadecimal. If no argument is given, a list of the structure names for which sizes are available is printed.

`sndd` [`–e`] [`–f`] [`–w` *file*] [[`–p`] *table_entry* ...]
Print the send descriptor table.

`sni_debug` [*value*]
Set the value of *sni_debug_level* to *value*.

`snode` [`–e`] [`–f`] [`–w` *file*] [[`–p`] *table_entry* ...]
Print information about open special files.

`sperr` [`–w` *file*] [`–e`] [`–f`]
Abbreviation: `sp`.
Gives information and hardware exchange recommendations when a SP bus error occurs. This function is only supported during testing of the system storage dump of a crashed RM600-xxx.
Print the MMU segment table values.

`srmount` [`–w` *file*] [[`–p`] *table_entry* ...]
Print the server mount table.

`stack` [–w *file*] [–u] [*process*]
`stack` [–w *file*] [–k] [*process*]
`stack` [–w *file*] [[–p] –i *start_addr*]
Alias: `s`.

Dump the stack. The (`–u`) option prints the user stack. The (`–k`) option prints the kernel stack. The (`–i`) option prints the interrupt stack starting at *start_addr*. If no arguments are entered, the kernel stack for the current process is printed. The interrupt stack and the stack for the current process are not available on a running system.

`stat` [–w *file*]
Print system statistics.

`stream` [–e] [–f] [–w *file*] [[–p] *table_entry* ...]
Print the streams table.

`strstat` [–w *file*]
Print streams statistics.

`strtty` [–w *file*] [–l] *strtty_addr*
Output the `struct strtty` data structure for the specified *strtty_addr* address.

`stt` [–w *file*] *stt_addr*
Output the data structure of type `stt_t` for the specified *stt_addr* address.

`tco` [–w *file*] [–f] [–t] [–p] *tco_addr* ...
Output the `struct tco_endpt` data structure for the specified *tco_addr* addresses.

`tcp_clear_buf` [–w *file*]
Delete the logged `tcp` information.

`tcp_set_faddr` *internet_addr*
Set the remote Internet address to be logged.

`tcp_set_laddr` *internet_addr*
Set the local Internet address to be logged.

`tcp_trace` [–w *file*]
Output the logged `tcp` information.

`tcp_trace_status` [–w *file*]
Show the status of the logging of the `tcp` information.

`tcpcb` [–w *file*] [–p] *tcpcb_addr* ...
Output the `tcp` administration structure (`struct tcpcb`) for the specified *tcpcb_addr* addresses.

`tcpiphdr` [–w *file*] [–p] *tcpiphdr_addr* ...
Abbreviation: `ti`.

Output the `struct tcpiphdr` data structure for the specified *tcpiphdr_addr* addresses.

`tcpstat` [–w *file*] [[–p] *tcpstat_addr* ...]
Show statistical information on `tcp`. If the *tcpstat_addr* address is specified, this data structure (`struct tcpstat`) is then used for that.

`tpipcb` [–w *file*] [–p] *tpipcb_addr* ...
Output the contents of the `tpipcb` structure for the specified addresses.

`tpioob` [–w *file*] [–p] [–l] *tpioob_addr* ...
Show the contents of the `tpioob` structures. If the `–l` option is specified, the list of `tpioob` structures is output beginning with the specified addresses.

`trace` [–w *file*] [–r] [*process*]
`trace` [–w *file*] [[–p] –I *start_addr*]
Alias: `t`.

Print stack trace. The kfp value is used with the `–r` option; the `kfp` function prints or sets the kfp (kernel frame pointer) value. The interrupt option prints a trace of the interrupt stack beginning at

start_addr. The interrupt stack trace and the stack trace for the current process are not available on a running system.

`troff`
Disable detailed output of the stack log.

`tron` Enable detailed output of the stack log.

`ts` [–w *file*] *virtual_address* ...
Print text symbol closest to the designated address.

`tsdptbl` [–w *file*] [*table_entry* ...]
Print the time-sharing scheduler parameter table [see `ts_dptbl(4)`].

`tsproc` [–w *file*]
Print information about processes in the time-sharing scheduler class.

`tty` [–e] [–f] [–l] [–w *file*] [–t*type* [[–p] *table_entry* ...]]
`tty` [–e] [–f] [–l] [–w *file*] [[–p] *start_addr*]
Valid types: `pp`, `iu`.

Print the tty table. If no arguments are given, the tty table for both tty types is printed. If the `–t` option is used, the table for the single tty type specified is printed. If no argument follows the type option, all entries in the table are printed. A single tty entry may be specified using *start_addr*. The `–l` option prints the line discipline information.

`udpstat` [–w *file*] [[–p] *udpstat_addr* ...]
Show statistical information on `udp`. If the *udpstat_addr* address is specified, this data structure (`struct udpstat`) is used for that.

`uinode` [–e] [–f] [–w *file*] [[–p] *table_entry* ...]
Alias: `ui`.

Print the ufs inode table.

`user` [–f] [–w *file*] [*process*]
Alias: `u`.

Print the `struct user` data structure for the designated process. This structure is referenced in the kernel by the variable `u` (i.e. `u.u_cdir`, `u.u_signal`).

`var` [–w *file*]
Alias: `v`.

Print the tunable system parameters.

`vfs` [–e] [–w *file*] [[–p] *table_entry* ...]
Alias: `mount`, `m`.

Print information about mounted file systems.

`vfssw` [–w *file*] [[–p] *table_entry* ...]
Print information about configured file system types.

`vnode` [–w *file*] [[–p] *vnode_addr* ...]
Print information about virtual inodes. A virtual inode provides a generic inode interface to the system. It is the part of VFS (virtual file system) that represents a generic file system to the base system. Each virtual inode has a file system specific data area that the internal file system (e.g. `ufs`) uses for its on-disk inode.

Outside a particular file system, only virtual inodes are used and operated on. For instance, the current directory of a process is represented in the user area (`u.u_cdir`) as a pointer to a virtual inode.

`vtop` [–w *file*] [–s *process*] *start_addr* ...
Print the physical address translation of the virtual address *start_addr*.

`w` [–f] [–w *file*] {*addr value*} ...
Write the *value* to *addr*.

`zx` [–w `file`] [[–i] *interface_id*] [[–r] *interface_id*] [[–t] *interface_id*] [[–m] *interface_id*] \

[[–C] *interface_id*]

Output information about the Ethernet LAN interfaces handled by the `zx` driver. If no arguments are specified when the command is called, all available `zx` interfaces and important global variables of the `zx` driver are output.

The structures of the specified interface are output with the `–i` option.

The contents of the receive buffer are output if the `–r` option is specified.

The contents of the send buffer are output if the `–t` option is specified.

The `–m` option outputs MIB related data and an error statistic for the interface.

A counter statistic is output for the interface with the `–C` option.

Functions in ikdb emulation mode

Following the switch to `ikdb` emulation mode with the `crash` function, all functions known to `ikdb(8)` are available. However, there are some functions that are not contained in `crash`:

`arp`, `assert`, `B`, `bp`, `call`, `D`, `d`, `dump`, `ki`, `lb`, `mm`, `more`, `pa`, `rnode`, `rtable`, `set`, `shortio`, `siminfo`, `so`, `st`, `strtty`, `tco`, `tim_tim`, `un`, `W`, `w`, `wa`.

Additional functions in ikdb emulation mode

`cd` [*directory_path*]
: Set path for structure module.

`devno` *addr*
: Output the major and minor number of the device.

`file` [–w] [*filename*]
: Activate output redirect for `ikdb` emulation. If the `–w` option is specified, output is only sent to *filename*, otherwise it is sent to standard output and to the file. The `file` function without parameters resets the output redirect.

`ia4` *addr*
: Show the contents of a specific f4bsd44-in_ifaddr structure.

`ifa4` *addr*
: Show the contents of a specific `bsd44-ifaddr` structure.

`ifp6` *addr*
: Show the contents of a specific `bsd44-ifp` structure.

`inpcb` [*address* | `offset`]
: Show the contents of a specific `inpcb` structure or a list of all `inpcb` structures.

`inpcb6` *addr*
: Show the contents of a specific `bsd44-inpcb` structure.

`kmapool` *kmp_addr*
: Show the contents of a specific `kmp` structure.

`pr_abic` *addr*
: Show the contents of the IFC/BIC folder after *addr*. This function is only supported for the RM600-xxx.

`ps` [*address*] [*type*]
: Show the contents of a specific structure for *address*. The structure here is of type *type*, and must be defined as a structure module.

`rn` *addr*
: Show the contents of a specific `radix` structure.

`rnwalk` *addr*
: Show the routing tree from the specified node.

`route6` [*addr*]
: Output all or specific (*addr*) data structures of type `route`.

`rtent6` *addr*
Show the contents of a specific `bsd44-rtentry` structure.

`socket6` [*addr*]
Show all or specific (*addr*) `socket` structures.

`tcpcb` {*address* | `offset`}
Show the contents of a specific `tcpcb` structure.

`tcpcb6` *addr*
Show the contents of a specific `bsd44-socket` structure.

`tpi6` *addr*
Show the contents of a specific `bsd44-tpi` structure.

`troff`
Disable detailed output of stack protocol.

`tron` Enable detailed output of stack protocol.

`vtop` *proc vaddr*
Show stages in converting virtual to physical storage addresses for the *proc vaddr* address pair.

`vxfset` *addr*
Show the contents of a specific `vx_fset` structure.

`vxrwsleep` *addr*
Show the contents of a specific `rwsleep` structure.

`vxsleep` *addr*
Show the contents of a specific `sleep` structure.

FILES

`/dev/mem`
system image of currently running system

SEE ALSO

`ep`(7), `ikdb`(8).

System Administrator's Reference Manual

cron(1M)

NAME

cron - clock daemon

SYNOPSIS

/usr/sbin/cron

DESCRIPTION

The cron command starts a process that executes commands at specified dates and times. Regularly scheduled commands can be specified according to instructions found in crontab files in the /var/spool/cron/crontabs directory. Users can submit their own crontab file via the crontab(1) command. Commands which are to be executed only once may be submitted via the at(1) command.

cron only examines crontab files and at command files during process initialization and when a file changes via the crontab or at commands. This reduces the overhead of checking for new or changed files at regularly scheduled intervals.

Since cron never exits, it should be executed only once. This is done routinely through /etc/rc2.d/S75cron when the system boots. /etc/cron.d/FIFO is used as a lock file to prevent more than one cron being executed.

To keep a log of all actions taken by cron, CRONLOG=YES (by default) must be specified in the /etc/default/cron file. If CRONLOG=NO is specified, no logging is performed. Keeping the log is a user configurable option since cron usually creates huge log files.

To specify the maximum number of executable jobs, configure MAXRUN=25 (default) to a value between 10 and 5120 in the /etc/default/cron file.

NOTES

cron takes account of the changeover from winter time to summer time, i.e. cron recognizes the change of time and reinitializes when the changeover takes place.

Correct recognition of the changeover from winter to summer time and vice versa means that when the clock is put back, cron routines are executed twice, and when the clock is put forward, cron routines are skipped. Therefore, if possible, no cron routines should be entered at the time when the clock is being put forward or back.

If values from the /etc/default/cron file are changed while the system is running, these changed values only take effect once cron is restarted [see rc2(1M)].

DIAGNOSTICS

A history of all actions taken by cron is recorded in /var/cron/log.

LOCALE

The LC_MESSAGES environment variable governs the language in which message texts are displayed.

The LC_TIME environment variable governs the format of date and time strings.

If LC_MESSAGES or LC_TIME is undefined or is defined as the null string, it defaults to the value of LANG. If LANG is likewise undefined or null, the system acts as if it were not internationalized.

If any of the locale variables has an invalid value, the system acts as if none of the variables was set.

The LC_ALL environment variable governs the entire locale. LC_ALL takes precedence over all the other environment variables that affect internationalization.

FILES

/etc/cron.d
: main cron directory

`/etc/default/cron`
used to enable/disable the logfile

`/etc/cron.d/FIFO`
used as a lock file

`/var/cron/log`
accounting information

`/var/spool/cron`
spool area

SEE ALSO

`at(1)`, `crontab(1)`, `sh(1)`, `cron(4)`, `queuedefs(4)`.

daps(1M) - erruser(1M)

System Administrator's Reference Manual

daps(1M)

NAME

`daps` - Autologic APS-5 typesetter filter

SYNOPSIS

`daps` [*option* ...] [*file* ...]

DESCRIPTION

`daps` prints *files* created by `troff(1M)` on any printer that accepts the APS-5 graphics and page layout language. If no *file* is mentioned, the standard input is printed.

OPTIONS

`-f` *dir*
Uses *dir* as the font directory (default for *dir* is `/usr/lib/font/devaps`).

`-F` *dir*
Same as `-f`.

`-t` Uses standard output (if you are not going to have `daps` directly drive the APS-5 typesetter).

`-r` Reports the number of pages.

`-A` Writes the accounting information to stderr.

`-b` Reports whether typesetter is busy or not. Nothing is printed.

`-w` Waits until typesetter is available, then processes the job.

`-o` *list*
Only processes the pages whose numbers are given in the comma-separated *list*. The list may contain single pages or page ranges, where the latter consists of a pair of pages separated by "-".

`-s` *num*
Stops processing every *num* pages, and halts the typesetter.

`-v` *num*
Uses *num* as the maximum vertical step size up or down the page. The argument *num* is interpreted as tenths of a point.

`-h` *str*
Uses *str* as the string to be printed in the header.

`-H` *path*
Uses *path* as the pathname of the file whose first line contains the string to be printed in the header.

`-d` *number*
: Toggles the debug flags for each number contained in the string *number*. If *number* contains the character `"*"` then all of the debug flags will be toggled. Maximum for *number* is 20.

`-D` *dump_file*
: Writes all of the debug information into file *dump_file*.

`-L` *log_file*
: Writes all error messages into file *log_file*.

`-I`
: Ignores all fatal errors. This option is only to be used for debugging – it may cause a core file to be written.

The input file *file* must be generated by `troff`. Best results will almost certainly be obtained only when `troff` and `daps` agree on the target printer.

EXAMPLES

1) Output directly to the APS-5 phototypesetter:

`pic` *file* | `eqn` | `troff -mm` | `daps`

2) Redirect output from standard output to the APS-5 phototypesetter:

`pic` *file* | `eqn` | `troff -mm` | `daps -t` | `lp -d`*aps-printer*

FILES

`/usr/lib/font/devaps`

SEE ALSO

`troff`(1M), `troff`(5).

System Administrator's Reference Manual

dcopy(1M)

NAME

`dcopy` - copy file systems for optimal access time

SYNOPSIS

`dcopy` [`-F` *FSType*] [`-V`] [*generic_options*] [`-o` *specific_options*] *inputfs outputfs*

DESCRIPTION

`dcopy` copies file system *inputfs* to *outputfs*. *inputfs* is the device file for the existing file system whereas *outputfs* is the device file to hold the reorganized result. For the most effective optimization, *inputfs* should be the raw device and *outputfs* should be the block device. Both *inputfs* and *outputfs* should be unmounted file systems.

There are two types of option: *generic_options* and *specific_options*. *generic_options* indicate options that are supported not only by the generic but also by the *FST*-specific module of the command (if that command exists). *specific_options* indicate suboptions that are supported only by the *FST*-specific module of the command. They are specified in a comma-separated list and/or as keyword-attribute pairs for interpretation by the *FST*-specific module of the command. These suboptions are specified via the `-o` option.

OPTIONS

The *generic_options* are:

`-F` Specifies the *FSType* on which to operate. The *FSType* should either be specified here or should be determined from `/etc/vfstab` by matching the *inputfs* (device) with an entry in the table.

`-V` Echoes the complete command line, but does not execute the command. The command line is generated using the options and arguments provided by the user, supplemented by information derived from `/etc/vfstab`. This option should be used to verify and validate the command line. When used with the *FSType* option, it passes the `-V` to the *FSType*-specific module for interpretation.

`-o` Specifies *FSType*-specific options.

NOTES

This command may not be supported for all FSTypes.

FILES

`/etc/vfstab`
List of default parameters for each file system

SEE ALSO

`vfstab`(4) .

System Administrator's Reference Manual

deact_ctlr(1M)
RM400 only

NAME

`deact_ctlr, set_ctlr_attr -` deactivate/set controller attributes

SYNOPSIS

`deact_ctlr –O|I|E|P –s` *slot_nr* [`–n` *channel*] [`–i` *id*] [`–e` *err*] [`–m` *comment*]

`set_ctlr_attr –O|I|E|P [ –A`*attr*] `–s` *slot_nr* [`–n` *channel*] [`–i`*id*] [`–e` *err*] [`–m` *comment*]

DESCRIPTION

The `deact_ctlr` command first checks whether a command is specified for this board type in the table `ctlr_defect.tab`, with which the board must be set to `DEFECT`. If this is the case, the command is executed.

The `DEFECT` flag in the configuration object `HW_CONFIG` is then set. The specified values are set here.

Activity on the board is then stopped as far as possible. The `ctlr_deact.tab` table defines whether activity can be stopped and how this happens. If a deactivation command is configured in this table, and the command was successfully executed, the `DEACTIVATE` flag in the configuration object `HW_CONFIG` is set with the corresponding values.

Using the `set_ctlr_attr` command, it is also possible to specify the flag to be processed by the command (1 = `DEFECT`, 2 = `ERROR`). If 1 (`DEFECT`) is specified, the command is identical to the `deact_ctlr` command. If 2 (`ERROR`) is specified, only the `ERROR` flag in `HW_CONFIG` is set with the corresponding values. Otherwise, no actions are executed.

OPTIONS

`–O` Onboard bus.

`–I` ISA bus.

`–E` EISA bus.

`–P` PCI bus.

`–A` *attr*
Specification of the attribute number (1 or 2). Default: 2.

`–s` *slot_nr*
Specification of the controller slot number.

`–n` *channel*
Specification of the controller channel number.

`–i` *id*
ID to be entered in the configuration object. Default: 0.

`–e` *err*
Error code to be entered in the configuration object. Default: 0.

`–m` *comment*
Comment to be entered in the configuration object.
Default: id = 0:
"deactivated by operator" or "set by operator"
Default: id = 1:
""
Default: id = 2:
"deactivated by LAR" or "set by LAR"

Specification of a channel (`-n` *channel*) is currently ignored, with the exception of CPU boards and MBP-HD, and the action is executed for the entire board.

RETURN VALUES

If no error occurs, a 0 is returned. The other return values have the following meanings:

`1 =` Invalid parameter or option

`2 =` – free –

`3 =` – free –

`4 =` The controller is not available in the system

`5 =` Error when setting the `DEFECT` flag in `HW_CONFIG`

`6 =` Error when setting the `ERROR` flag in `HW_CONFIG`

`7 =` Error when setting the `DEACTIVE` flag in `HW_CONFIG`

`8 =` Error when deleting the `DEFECT` flag in `HW_CONFIG`

`9 =` Error when deleting `ERROR` flag in `HW_CONFIG`

`10 =` Error when deleting the `DEACTIVE` flag in `HW_CONFIG`

`11 =` Error when deactivating the controller

`12 =` Error when activating the controller

FILES

```
/usr/bin/lar/ctlr_deact.tab
/usr/bin/lar/ctlr_defect.tab
/etc/init.d/lar_rc
/var/lar/hwconfig
```

SEE ALSO

`act_ctlr`(1M), `act_dev`(1M), `deact_dev`(1M), `lar`(7), `olr`(7).

System Administrator's Reference Manual

deact_dev(1M)

NAME

deact_dev, set_dev_attr - deactivate/set SCSI device attributes

SYNOPSIS

deact_dev [−i *id*] [−e *err*] [−m *comment*] *device_name*

set_dev_attr [−A *attr*] [−i *id*] [−e *err*] [−m *comment*] *device_name*

DESCRIPTION

The deact_dev command first checks whether the specified device is contained in the system configuration (autoconf). If it is not, the command is terminated with an error. If there is an entry in the system configuration (autoconf), the device is marked as DEFECT in the configuration object with the specified parameters. An attempt is then made to to remove the device from the system configuration using the changedev −f command.

Using the set_dev_attr command, it is also possible to specify the flag to be processed by the command (1 = DEFECT, 2 = ERROR). If 1 (DEFECT) is specified as the flag, the command is identical to the deact_dev command. If 2 (ERROR) is specified as the flag, only the ERROR flag in HW_CONFIG is set with the corresponding values. Otherwise, no actions are executed.

OPTIONS

−A *attr*
: Attribute number (1 or 2). Default: 2.

−i *id*
: ID to be entered in the configuration object. Default: 0.

−e *err*
: Error code to be entered in the configuration object. Default: 0.

−m *comment*
: Comment to be entered in the configuration object.

  Default: id = 0:
  : "deactivated by operator" or "set by operator"

  Default: id = 1:
  : ""

  Default: id = 2:
  : "deactivated by LAR" to "set by LAR"

device_name
: Name of the device as it is specified with autoconf −l (e.g. ios0/sdisk042).

RETURN VALUES

If no error occurs, 0 is returned. The other return values have the following meanings:

1 = Invalid parameter or option

2 = Error processing HW_CONFIG

3 = Error in changedev −f

WARNING

Components which were deactivated by LAR functions are not available to the user after the next reboot.

FILES

/sbin/scsilar_rc

/var/lar/hwconfig

SEE ALSO

act_dev(1M), reset_lar(1M), set_lar(1M), lar(7), olr(7), autoconf(8), changedev(8).

System Administrator's Reference Manual

delsysadm(1M)

NAME

`delsysadm -` sysadm interface menu or task removal tool

SYNOPSIS

`delsysadm` *task* | [`-r`] *menu*

DESCRIPTION

The `delsysadm` command deletes a *task* or *menu* from the `sysadm` interface and modifies the interface directory structure on the target machine.

OPTIONS

task | *menu*

The logical name and location of the menu or task within the interface menu hierarchy. Begin with the top menu `main` and proceed to where the menu or the task resides, separating each name with colons (see **EXAMPLES**).

`-r` If the `-r` option is used, this command will recursively remove all submenus and tasks for this menu. If the `-r` option is not used, the menu must be empty.

`delsysadm` should only be used to remove items added as "online" changes with the `edsysadm` command. Such an addition will have a package instance tag of ONLINE. If the task or menu (and its submenus and tasks) have any package instance tags other than ONLINE, you are asked whether to continue with the removal or to exit. Under these circumstances, you probably do not want to continue and you should rely on the package involved to take the necessary actions to delete this type of entry.

The command exits successfully or provides the error code within an error message.

EXAMPLES

To remove the `nformat` task, execute:

```
delsysadm main:applications:ndevices:nformat.
```

DIAGNOSTICS

`0` Successful execution

`2` Invalid syntax

`3` Menu or task does not exist

`4` Menu not empty

`5` Unable to update interface menu structure

NOTES

Any menu that was originally a placeholder menu (one that only appears if submenus exist under it) will be returned to placeholder status when a deletion leaves it empty.

When the `-r` option is used, `delsysadm` checks for dependencies before removing any subentries. (A dependency exists if the menu being removed contains an entry placed there by an application package). If a dependency is found, the user is shown a list of packages that depend on the menu being deleted and asked whether or not to continue. If the answer is yes, the menu and all of its menus and tasks are removed (even those shown to have dependencies). If the answer is no, the menu is not deleted.

`delsysadm` should only be used to remove menu or task entries that have been added to the interface with `edsysadm`.

SEE ALSO

edsysadm(1M), sysadm(1M).

System Administrator's Reference Manual

deluser(1M)

NAME

`deluser` - delete a user's login from the system

SYNOPSIS

`/sbin/deluser` *login* `Yes` | `No` *home_directory*

DESCRIPTION

The `deluser` command removes a user from the system by deleting the user's *login* from the `etc/passwd` and `etc/shadow` files. If `Yes` is specified as the second argument, the user's mailbox is removed and the user's home directory, including all files, is moved to `lost+found/`*login*.

NOTES

Recommend using `userdel(1M)` instead of `deluser`.

FILES

`/etc/passwd`
`/etc/shadow`
`/usr/mail`
`/lost+found`
`/etc/.useradm`

SEE ALSO

`adduser(1M)`, `passmgmt(1M)`, `userdel(1M)`.

System Administrator's Reference Manual

devattr(1M)

NAME

`devattr` - lists device attributes

SYNOPSIS

`devattr` [`–v`] *device* [*attribute* [...]]

DESCRIPTION

`devattr` displays the values for a device's attributes. The display can be presented in two formats. Used without the `–v` option, only the attribute values are shown. Used with the `–v` option, the attributes are shown in an *attribute=value* format. When no attributes are given on the command line, all attributes for the specified device are displayed in alphabetical order by attribute name. If attributes are given on the command line, only those are shown and they are displayed in command line order.

OPTIONS

`–v` Specifies verbose format. Attribute values are displayed in an *attribute=value* format.

device
Defines the device whose attributes should be displayed. Can be the pathname of the device or the device alias.

attribute
Defines which attribute, or attributes, should be shown. Default is to show all attributes for a device. See the `putdev`(1M) manual page for a complete listing and description of available attributes.

DIAGNOSTICS

The command will exit with one of the following values:

`0` Successful completion of the task.

`1` Command syntax incorrect, invalid option used, or internal error occurred.

`2` Device table could not be opened for reading.

`3` Requested device could not be found in the device table.

`4` Requested attribute not defined for specified device.

FILES

`/etc/device.tab`

SEE ALSO

`putdev`(1M) .

System Administrator's Reference Manual

devfree(1M)

NAME

`devfree` - release devices from exclusive use

SYNOPSIS

`devfree` *key* [*device* [...]]

DESCRIPTION

`devfree` releases devices from exclusive use. Exclusive use is requested with the `devreserv` command.

When `devfree` is invoked with only the *key* argument, it releases all devices that have been reserved for that *key*. When called with *key* and *device* arguments, `devfree` releases the specified devices that have been reserved with that *key*.

The arguments for this command are:

key Designates the unique key on which the device was reserved.

device
Defines the device that this command will release from exclusive use. Can be the pathname of the device or the device alias.

DIAGNOSTICS

The command will exit with one of the following values:

`0` Successful completion of the task.

`1` Command syntax incorrect, invalid option used, or internal error occurred.

`2` Device table or device reservation table could not be opened for reading.

`3` Reservation release could not be completely fulfilled because at least one of the devices was not reserved, or was not reserved on the specified key.

NOTES

The commands `devreserv` and `devfree` are used to manage the availability of devices on a system. These commands do not place any constraints on the access to the device. They serve only as a centralized bookkeeping point for those who wish to use them. Processes that do not use `devreserv` can simultaneously use a device with a process that has reserved that device.

FILES

`/etc/device.tab`
`/etc/devlkfile`

SEE ALSO

`devreserv`(1M) .

System Administrator's Reference Manual

devnm(1M)

NAME

`devnm` - device name

SYNOPSIS

`/usr/sbin/devnm` [*name* ...]

DESCRIPTION

The `devnm` command identifies the special file associated with the mounted file system where the *name* argument resides. One or more *name*s can be specified.

This command is most commonly used by the `brc` command to construct a mount table entry for the `root` device.

EXAMPLES

The command:

```
/usr/sbin/devnm /usr
```

produces:

```
/dev/ios0/rsdisk000s3  /usr
```

if `/usr` is mounted on `/dev/ios0/rsdisk000s3`.

LOCALE

The `LC_MESSAGES` environment variable governs the language in which message texts are displayed.

If `LC_MESSAGES` is undefined or is defined as the null string, it defaults to the value of `LANG`. If `LANG` is likewise undefined or null, the system acts as if it were not internationalized.

If any of the locale variables has an invalid value, the system acts as if none of the variables was set.

The `LC_ALL` environment variable governs the entire locale. `LC_ALL` takes precedence over all the other environment variables that affect internationalization.

FILES

```
/dev/ios*/*disk*
/dev/ipi*/*disk*
/etc/mnttab
```

SEE ALSO

`brc`(1M), `mnttab`(4).

System Administrator's Reference Manual

devreserv(1M)

NAME

`devreserv -` reserve devices for exclusive use

SYNOPSIS

`devreserv` [*key* [*devicelist* [...]]]

DESCRIPTION

`devreserv` reserves devices for exclusive use. When the device is no longer required, use `devfree` to release it.

`devreserv` reserves at most one device per *devicelist*. Each list is searched in linear order until the first available device is found. If a device cannot be reserved from each list, the entire reservation fails.

When `devreserv` is invoked without arguments, it lists the devices that are currently reserved and shows under which key it was reserved. When `devreserv` is invoked with only the *key* argument, it lists the devices that are currently reserved to that key.

The arguments for this command are:

key Designates a unique key on which the device will be reserved. The key must be a positive integer.

devicelist
Defines a list of devices that `devreserv` will search to find an available device. (The list must be formatted as a single argument to the shell.)

EXAMPLES

To reserve a disk drive and a tape drive:

```
$ key=$$
$ echo "The current Process ID is equal to: $key"
The Current Process ID is equal to: 10658
$ devreserv $key disk1 qtape1
```

To list all devices currently reserved:

```
$ devreserv
disk1         2423
disk2         10658
qtape1        10658
```

To list all devices currently reserved to a particular key:

```
$ devreserv $key
disk1
qtape1
```

DIAGNOSTICS

The command will exit with one of the following values:

`0` Successful completion of the task.

`1` Command syntax incorrect, invalid option used, or internal error occurred.

`2` Device table or device reservation table could not be opened for reading.

`3` Device reservation request could not be fulfilled.

NOTES

The commands `devreserv` and `devfree` are used to manage the availability of devices on a system. Their

use is on a participation basis and they do not place any constraints on the actual access to the device. They serve as a centralized bookkeeping point for those who wish to use them. To summarize, devices that have been reserved cannot be used by processes which utilize the device reservation functions until the reservation has been canceled. However, processes that do not use device reservation may use a device that has been reserved since such a process would not have checked for its reservation status.

FILES

`/etc/devlkfile`

SEE ALSO

`devfree`(1M) .

System Administrator's Reference Manual

di10(1M)

NAME

`di10` - Imagen printer filter

SYNOPSIS

`di10` [*option* ...] [*file* ...]

DESCRIPTION

`di10` prints *files* created by `troff(1M)` on any printer that accepts the Impress graphics and page layout language. If no *file* is mentioned, the standard input is printed. The following options are understood:

−o*list*
: Prints pages whose numbers are given in the comma-separated *list*. The list contains single numbers *N* and ranges *N*1 − *N*2. A missing *N*1 means the lowest-numbered page, a missing *N*2 means the highest.

−x*num*
: Leaves a margin *num* inches from the left side edge of the paper.

−y*num*
: Leaves a margin *num* inches down from the top of the page.

−l
: Prints the job in landscape mode.

−p*num*
: Sets the horizontal and vertical step sizes for graphics to *num* pixels. If Impress graphics commands are used for drawing this option will only affect splines.

−P*num*
: Uses *num* as the pixel diameter of the Impress graphics pen. Only useful when Impress graphics commands are being used.

−T*name*
: Generates output to be printed on the target printer *name*. The only supported choice for *name* is `i10`.

−F*dir*
: Uses *dir* as the `troff` font directory.

−B*dir*
: Uses *dir* as the raster table directory.

−R*name*
: Uses raster tables generated for device *name* when printing *file*. By default `di10` will attempt to use the target printer's raster files.

The *files* should be prepared by `troff`. Best results will almost certainly be obtained only when `troff` and `di10` agree on the target printer.

FILES

```
/usr/lib/font/devi10
/usr/lib/font/devi10/rasti10
```

SEE ALSO

`troff(1M)`, `troff(5)`.

System Administrator's Reference Manual

diffmk(1M)

NAME

`diffmk` - mark differences between versions of an input file

SYNOPSIS

`diffmk` *oldfile newfile markedfile*

DESCRIPTION

`diffmk` compares two versions of a file and creates a third file that includes "change mark" requests (`.mc`) for `nroff`(1M) or `troff`(1M). *oldfile* and *newfile* are the old and new versions of the file. `diffmk` generates *markedfile*, which, contains the text from *newfile* with "change mark" requests (`.mc`) inserted where *newfile* differs from *oldfile*. When *markedfile* is formatted, changed or inserted text is shown by a | at the right margin of each line. The position of deleted text is shown by a single asterisk: `*`.

`diffmk` can also be used in conjunction with the proper `nroff`/`troff` requests to produce program listings with marked changes. In the following command line:

```
diffmk old.c new.c marked.c ; nroff reqs marked.c | pr
```

the file `reqs` contains the requests

```
.pl 1
.ll 77
.nf
.eo
```

which eliminate page breaks (`.pl`), adjust the line length (`.ll`), set no-fill mode and turn off hyphenation (`.nf`), and ignore escape characters (`.eo`), respectively.

If the characters | and `*` are inappropriate, you might run *markedfile* through `sed`(1) to globally change them.

NOTES

Aesthetic considerations may dictate manual adjustment of some output. File differences involving only formatting requests may produce undesirable output, i.e. replacing `.sp` by `.sp 2` will produce a "change mark" on the preceding or following line of output.

The command `/usr/bin/diffmk` is identical to the UCB version `/usr/ucb/diffmk`.

SEE ALSO

`diff`(1), `nroff`(1M), `troff`(1M).

System Administrator's Reference Manual

dispadmin(1M)

NAME

dispadmin - process scheduler administration

SYNOPSIS

dispadmin –l

dispadmin –c *class* –g [–r *res*]

dispadmin –c *class* –s *file*

DESCRIPTION

The dispadmin command displays or changes process scheduler parameters while the system is running.

The –l option lists the scheduler classes currently configured in the system.

The –c option specifies the class whose parameters are to be displayed or changed. Valid *class* values are RT for the real-time class, TS for the time-sharing class, and FP for the fixed-priority class.

The –g option gets the parameters for the specified class and writes them to the standard output. Parameters for the real-time class are described in rt_dptbl(4) . Parameters for the time-sharing class are described in ts_dptbl(4) . Parameters for the fixed-priority class are described in fp_dptbl(4) .

When using the –g option you may also use the –r option to specify a resolution to be used for outputting the time quantum values. If no resolution is specified, time quantum values are in milliseconds. If *res* is specified it must be a positive integer between 1 and 1,000,000,000 inclusive, and the resolution used is the reciprocal of *res* in seconds. For example, a *res* value of 10 yields time quantum values expressed in tenths of a second; a *res* value of 1,000,000,000 yields time quantum values expressed in microseconds. If the time quantum cannot be expressed as an integer in the specified resolution, it is rounded up to the next integral multiple of the specified resolution.

The –s option sets scheduler parameters for the specified class using the values in *file*. These values overwrite the current values in memory – they become the parameters that control scheduling of processes in the specified class. The values in *file* must be in the format output by the –g option. Moreover, the values must describe a table that is the same size (has same number of priority levels) as the table being overwritten. Superuser privileges are required in order to use the –s option.

The –g and –s options are mutually exclusive: you may not retrieve the table at the same time you are overwriting it.

dispadmin does some limited sanity checking on the values supplied in *file* to verify that they are within their required bounds. The sanity checking, however, does not attempt to analyze the effect that the new values have on the performance of the system. Inappropriate values can have a dramatic negative effect on system performance.

EXAMPLES

The following command retrieves the current scheduler parameters for the real-time class from kernel memory and writes them to the standard output. Time quantum values are in microseconds.

```
dispadmin –c RT –g –r 1000000
```

The following command overwrites the current scheduler parameters for the real-time class with the values specified in rt.config.

```
dispadmin –c RT –s rt.config
```

The following command retrieves the current scheduler parameters for the time-sharing class from kernel memory and writes them to the standard output. Time quantum values are in nanoseconds.

```
dispadmin –c TS –g –r 1000000000
```

The following command overwrites the current scheduler parameters for the time-sharing class with the values

specified in `ts.config`.

```
dispadmin –c TS –s ts.config
```

The following command retrieves the current scheduler parameters for the fixed-priority class from kernel memory and writes them to the standard output. Time quantum values are in nanoseconds.

```
dispadmin –c FP –g –r 1000000000
```

The following command overwrites the current scheduler parameters for the fixed-priority class with the values specified in `fp.config`.

```
dispadmin –c FP –s fp.config
```

DIAGNOSTICS

`dispadmin` prints an appropriate diagnostic message if it fails to overwrite the current scheduler parameters due to lack of required permissions or a problem with the specified input file.

SEE ALSO

`priocntl`(1), `priocntl`(2), `fp_dptbl`(4), `rt_dptbl`(4), `ts_dptbl`(4).

System Administrator's Reference Manual

dispgid(1M)

NAME

`dispgid -` display a list of all valid group names

SYNOPSIS

```
dispgid
```

DESCRIPTION

`dispgid` displays a list of all group names on the system (one group per line).

EXIT CODES

`0` = Successful execution

`1` = Cannot read the group file

System Administrator's Reference Manual

dispuid(1M)

NAME

`dispuid -` display a list of all valid user names

SYNOPSIS

```
dispuid
```

DESCRIPTION

`dispuid` displays a list of all user names on the system (one line per name).

EXIT CODES

`0` = Successful execution

`1` = Cannot read the password file

System Administrator's Reference Manual

dkstat(1M)

NAME

`dkstat -` report magnetic disk I/O statistics

SYNOPSIS

`dkstat` [`–abcdgpqrstvx`] [*dkname ...*]

`dkstat –z` [*dkname ...*]

`dkstat` *interval* [*count*] [*dkname ...*]

`dkstat –L` *interval* [*count*] [*dkname ...*]

DESCRIPTION

`dkstat` outputs various I/O statistics collected for disk-based I/O devices since the system was loaded or since the last time the I/O statistic was reset.

Disk-based I/O devices include:

- Physical hard disks (`sdisk`)
- RAID Luns (`sraid`)
- Virtual disks (`vdisk`)

Scaled profiles can be displayed for the block size, the length of the disk queue, access times, cylinder and partition usage, disk response times, sort queue times, response time characteristics as well as general details about average utilization and total usage.

Current disk usage can be monitored by specifying an *interval* in seconds. A continuous display of average usage over each *interval* is produced.

You must select an option or specify an *interval*. `dkstat` will not work correctly without an option or *interval*.

Statistics can be cleared with the `–z` flag. If you use the `–z` option with other options, the other options will be ignored; therefore, always use the `–z` option on a separate line.

OPTIONS

`–a` All statistical profiles. □

`–b` A profile of the number of blocks that were requested in each I/O. □

`–c` Profile of the cylinders on which I/O occurred. □

`–d` Profile of the response time in milliseconds from the sending of the I/O to the device and the receipt of the completion interrupt. □

`–g` Read, write and byte counts. The read and write counts are the total number of requests of each type. The byte count is the total number of bytes transferred in both read and write requests. □

`–p` Profile of the disk partitions on which I/O occurred. □

`–q` Profile of the disk request queue length at the time each request is added to the queue. □

`–r` Profile of the response time in milliseconds from the issue of the I/O to the receipt of the completion interrupt. This is the sum of the values displayed by the `–d` and `–t` flags.

`–s` Profile of the absolute value of the distance sought in cylinders.

`–t` Profile of the response time in milliseconds from the time the I/O is added to the queue to the time the I/O is sent to the device.

`–v` Forces the display of all data. If the `–v` flag is not used, `dkstat` will truncate trailing zeros and print ellipses at the end of non-zero data.

`–x` Prints a summary of the options.

`-z` Clears all statistics.

`-L` Display load average over interval.

The optional trailing *interval* causes `dkstat` to report the average disk transfers per second on each disk over that *interval* repeated an optional *count* times. If *count* is omitted, the output is repeated continuously. Drives may be specified with an optional trailing *dkname*. Names are given in either the 13 character device name form or an abbreviated form which includes bus number, bus type, and controller and device numbers. For example, `ios0/sdisk001` or `s001`. The shell metacharacter syntax "`?`", "`*`", and "`[`" may also be used. For example, `ios0/sdisk* ?disk[01]??`, or even `*[01]`. These names should be escaped on shell level where necessary.

If no drives are specified, then the five disks with the highest usage since boot will be displayed. If the `-L` load average flag is specified, then the four disks with highest usage during the interval are displayed.

Profile information is collected from several arrays within the kernel address space. Each array has two logical areas with selectable scaling of data. These arrays are described by maximum values of both segments and the scale factor within both segments.

Averages are generated which are based on scaling data with the beginning value of the sample cell or the end value of the sample cell. The true average can be extrapolated between these values. Because the data is collected in cells, the exact average cannot be determined.

The load average is a number indicating the overall effect of a disk on system I/O performance. While the details of I/O performance may be obtained by carefully examining the profile information, the load average is meant to give an overview of performance of each device in a single number. The disk load average is calculated as an exponential reduction of the current load. The load is calculated for an interval using the following formula:

```
load = device_busy_time * ( L1 + L2 )
```

where:

```
L1 = new_requests/requests_completed
L2 = average_queue_depth
```

SEE ALSO

`sar(1)`, `mpstat(1M)`, `vmstat(1M)`.

System Administrator's Reference Manual

download(1M)
Line Printer PostScript Utilities

NAME

`download` - host resident PostScript font download

SYNOPSIS

`/usr/lib/lp/postscript/download` [*options*] [*files*]

DESCRIPTION

`download` prepends host resident fonts to *files* and writes the results on standard output. If no *files* are specified, or if – is one of the input *files*, standard input is read. `download` assumes the input *files* are a single PostScript job and that requested fonts can be included at the start of each input *file*.

OPTIONS

`–f` Forces a complete scan of each input *file*. In the absence of an explicit comment pointing `download` to the end of the file, the default scan stops immediately after the PostScript header comments.

`–m`*name*
Uses *name* as the font map table. A *name* that begins with / is the full pathname of the map table. Otherwise *name* is relative to the host font directory.

`–p`*printer*
Reads the printer resident font list from file `/etc/lp/printers/`*printer*/residentfonts. Fonts named in this file are printer resident and will not be downloaded.

`–H`*dir*
Uses *dir* as the host font directory. The default is `/usr/share/lib/hostfontdir`.

`–T`*dir*
Uses *dir* as the temporary file directory. Only used to make a copy of standard input. By default *dir* is set to `/tmp`.

Requested fonts are named in a `%%DocumentFonts:` comment in the input *files*. Available fonts are the ones listed in the map table selected using the `–m` option.

The map table consists of fontname-filename pairs. The fontname is the full name of the PostScript font, exactly as it would appear in a `%%DocumentFonts:` comment. The filename is the pathname of the host resident font. A filename that begins with a / is used as is, otherwise the pathname is relative to the host font directory. Comments are introduced by `%` (as in PostScript) and extend to the end of the line.

The only candidates for downloading are fonts listed in the map table that point `download` to readable files. A font is downloaded at most once. Requests for unlisted fonts or inaccessible files are ignored. All requests are ignored if the map table can't be read.

EXAMPLES

A map table used to control the downloading of the Bookman font family might be,

```
% The first string is the full PostScript font name. The second
% string is the file name - relative the host font directory unless
% it begins with a /.
  Bookman-Light            KR
  Bookman-LightItalic      KI
  Bookman-Demi             KB
  Bookman-DemiItalic       KX
```

Use file myprinter (in the default host font directory) as the map table,

```
download -m myprinter file
```

Set the host font directory to `/tmp/font` and use `/tmp/font/xxx` as the map table,

```
download -H /tmp/font -mxxx file
```

DIAGNOSTICS

0exit status is returned if *files* were successfully processed.

NOTES

`download` should be part of a more general program.

`download` does not look for `%%PageFonts:` comments and there is no way to force multiple downloads of a particular font.

Using full pathnames, either in map tables or for the map table name, is not often recommended.

SEE ALSO

`dpost`(1M), `postdaisy`(1M), `postdmd`(1M), `postio`(1M), `postmd`(1M), `postprint`(1M), `posttek`(1M).

System Administrator's Reference Manual

dpost(1M)
Line Printer PostScript Utilities

NAME

`dpost` - troff postprocessor for PostScript printers

SYNOPSIS

`/usr/lib/lp/postscript/dpost` [*options*] [*files*]

DESCRIPTION

`dpost` translates *files* created by `troff`(1M) into PostScript and writes the results on standard output. If no *files* are specified, or if – is one of the input *files*, standard input is read.

OPTIONS

–c*num*
: Prints *num* copies of each page. By default only one copy is printed.

–e*num*
: Sets the text encoding level to *num*. The recognized choices are 0, 1, 2, and 3. The size of the output file and print time should decrease as *num* increases. The default is 2.

–m*num*
: Magnifies each logical page by the factor *num*. Pages are scaled uniformly about the origin, which is located near the upper left corner of each page. The default magnification is 1.0.

–n*num*
: Prints *num* logical pages on each piece of paper, where *num* can be any positive integer. By default *num* is set to 1.

–o*list*
: Prints pages whose numbers are given in the comma-separated *list*. The list contains single numbers *N* and ranges *N*1 – *N*2. A missing *N*1 means the lowest numbered page, a missing *N*2 means the highest.

–p*mode*
: Prints *files* in either **portrait** or **landscape** *mode*. Only the first character of *mode* (`p` or `l`) is significant. The default *mode* is portrait.

–w*num*
: Sets the line width used to implement `troff` graphics commands to *num* points, where a point is approximately 1/72 of an inch. By default *num* is set to 0.3 points.

–x*num*
: Translates the origin *num* inches along the positive x axis. The default coordinate system has the origin fixed near the upper left corner of the page, with positive x to the right and positive y down the page. Positive *num* moves everything right. The default offset is 0 inches.

–y*num*
: Translate the origin *num* inches along the positive y axis. Positive *num* moves text down the page. The default offset is 0.

–A*file*
: Appends a simple accounting record to *file* after all the input *files* have been successfully translated. By default no accounting data is produced.

–B
: Includes BoundingBox comments in the output file. Should only be used when the comments are needed for picture inclusion. Output is forced onto an 8.5 by 11.0 inch page.

–F*dir*
: Use *dir* as the font directory. By default *dir* is set to `/usr/lib/font`, and `dpost` reads ASCII font files from device directory `/usr/lib/font/devpost`.

–H*dir*
: Uses *dir* as the host resident font directory. Files in this directory should be complete PostScript font descriptions. File names must match the `troff` font names. There is no default directory.

–L*file*
: Uses *file* as the PostScript prologue, which by default is `/usr/lib/lp/postscript/dpost.ps`.

–T*name*
: Uses font files for device *name* as the best description of available PostScript fonts. By default *name* is set to `post` and `dpost` reads ASCII files from `/usr/lib/font/devpost`.

In addition, three options allow the insertion of arbitrary PostScript at controlled points in the translation process:

–C*file*
: Copies *file* to the output file. *file* follows the prologue but precedes any job initialization commands. *file* becomes part of the job's global environment and must contain legitimate PostScript commands.

–P*string*
: Adds *string* to the output file. *string* follows the prologue but precedes any job initialization commands. *string* becomes part of the job's global environment and must be legitimate PostScript.

–R*action*
: Requests special *action* (e.g. manualfeed) on a per page or global basis. The *action* string can be given as request, request:page, or request:page:file. If page is omitted or given as 0 the request applies to all pages. If file is omitted the request lookup is done in `/usr/lib/lp/postscript/ps.requests`. The collection of recognized requests can be modified or extended by changing `/usr/lib/lp/postscript/ps.requests`. Multiple occurrences of the `–R` option behave as expected.

The *files* should be prepared by `troff`. The default font files in `/usr/lib/font/devpost` produce the best and most efficient output. `dpost` makes no assumptions about resolutions. The first `x res` command sets the resolution used to translate the input *file*s, the DESC file, usually `/usr/lib/font/devpost/DESC`, sets the font table resolution, and the PostScript prologue is responsible for setting up an appropriate user coordinate system.

EXAMPLE

A typical command line might be,

```
pic file | tbl | eqn | troff -mm | dpost
```

WARNINGS

Output files will often be nonconforming, according to Adobe's file structuring conventions. Piping the output of `dpost` through `postreverse` should produce a minimally conforming PostScript file.

Although `dpost` can handle files formatted for any device, emulation is expensive and can easily double the print time and the size of the output file. No attempt has been made to implement the character sets or fonts available on all devices supported by `troff`. Missing characters will be replaced by white space, and unrecognized fonts will usually default to one of the Times fonts (i.e. `R`, `I`, `B`, or `BI`).

An `x res` command must precede the first `x init` command, and all the input *files* should have been prepared for the same output device.

Use of the `–T` option is not encouraged. Its only purpose is to enable the use of other PostScript font and device description files, that perhaps use different resolutions, character sets, or fonts.

DIAGNOSTICS

0exit status is returned if *files* were successfully translated, while 2 often indicates a syntax error in the input *files*.

FILES

`/usr/lib/font/devpost/*`

```
/usr/lib/font/devpost/charlib/*
/usr/lib/lp/postscript/dpost.ps
/usr/lib/lp/postscript/color.ps
/usr/lib/lp/postscript/draw.ps
/usr/lib/lp/postscript/forms.ps
/usr/lib/lp/postscript/ps.requests
```

SEE ALSO

postdaisy(1M), postdmd(1M), postio(1M), postmd(1M), postprint(1M), postreverse(1M), posttek(1M), troff(1M), font(5), troff(5).

System Administrator's Reference Manual

dptelog(1M)
RM400 only

NAME

`dptelog -` DPT Logger

SYNOPSIS

`dptelog –nowait` [`–pause=`{`off`|`on`}] [`–new`] [`–time=`*minutes*] [`–syslog=`{`off`|`on`}] [`–syslog:`*file-name*] [`–elog=`{`off`|`on`}] [`–notify=`*name*,*name*...] [`–notify=d:`*device*] [`–broadcast=`{`off`|`on`}] [`–midtemp=`{`off`|`on`}] [`–shelfok=`{`off`|`on`}] [`–mail`] [`–save`] [`–cmdtime=`*seconds*] [`–help`]

DESCRIPTION

`dptelog` is a "Broadcaster" for "Storage Manager". You may specify that event messages be sent to users, groups, devices, by e-mail, to the system log messages, and/or to the "Event Log File".

OPTIONS

`–nowait`
: Scans system once and exits.

`–pause=`{`off`|`on`}
: "Logger Initial Run Status".

`–new` Creates new "Event Log Files".

`–time=`*minutes*
: Once every specified "Time Interval", the Broadcaster will read the event logs of all DPT adapters. At this time, any new event messages will be broadcast to each specified destination.

`–syslog=`{`off`|`on`}
: If this option is enabled, events read from the adapters' event logs will be recorded in the "Event Log File" on hard disk. This ensures that events will not be lost when system is powered down. In addition, the "Event Log File" may contain a number of events limited only by the available disk space.

`–syslog=`*file-name*
: Outputs to "Event Log File" *file-name*; default is `/var/adm/log/dptelog.log`.

`–elog=`{`off`|`on`}
: Outputs to the "Event Log Files".

`–notify=`*name*,*name*...
: The name or list specifies which users and groups will receive event messages. User names and groups can be included in any desired combination. If the e-mail option is available and enabled then this list will also be used as destinations for e-mail containing the event messages.

`–notify=d:`*device*
: If broadcast is enabled the name or list specifies which devices will receive event messages.

`–broadcast=`{`off`|`on`}
: Broadcasts to specified users and/or to specified devices.

`–midtemp=`{`off`|`on`}
: Intermediate high temperature event occurring on the Host Bus Adapter (HBA) is stored with the data format in `/var/adm/log/eventlog.`*raid-id* if enabled.

  raid-id is a number indicating a RAID element of the "RAID ARRAY".

`–shelfok=`{`off`|`on`}
: Turns broadcasting on or off for the event relating to the peripherals cabinet. The event is any hardware fault such as the fan failure or power off. If enabled, the event is logged in `/var/adm/log/dptelog.log` and is sent to the output device (monitor) or to the user if specified.

`–mail`
Sends mail to specified users.

`–save`
Saves command line as default.

`–cmdtime=`*seconds*
Seconds before a command times out (default 20).

`–help`
Displays help screen.

FILES

`/var/adm/log/dptelog.log`
default log file

System Administrator's Reference Manual

dptmgr(1M)
RM400 only

NAME

`dptmgr` - SmartRAID Storage Manager

SYNOPSIS

`dptmgr [ –restart ]`

DESCRIPTION

`dptmgr` is a management and configuration tool for integrated RAID associations. Its functionality is as follows:

- It checks the hardware configuration when integrating the SmartRAID controller (CS31) and when adding on new SCSI devices.
- It creates, modifies, and deletes disk arrays.
- It provides online functions for the SCSI memory subsystems, for example, event logging and reporting, diagnostic functions, and I/O monitoring.

OPTIONS

`–restart`
: Terminates an active Storage Manager and starts a new one.

System Administrator's Reference Manual

dumpsave(1M)

NAME

dumpsave - save a system dump

SYNOPSIS

dumpsave [−c] [−q] [−d *dumpdir*] [−s *swapfile*] [−o]

DESCRIPTION

The dumpsave command should be called during the system boot process. Its function is to save a system dump from the swap device to a regular file.

dumpsave checks the swap device to see if a system dump exists. If one is found, it saves the system dump to the system dump directory, and also copies /unix there. If the system dump is saved successfully, the dump header is cleared from the swap device so that successive runs of dumpsave will not resave the same system dump. Both the dump and a copy of /unix are given an identical numeric suffix. The suffix used is the lowest number that doesn't already appear.

dumpsave initializes several parameters from /etc/default/dumpsave:

DUMPDIR=/var/crash
MINFREE=4096
UNIXFILE=/unix
SWAPFILE=/dev/swap
TMPDEV=*temporary-name-for-devices*
ULIMIT=*ulimit-value*

The first two, DUMPDIR and MINFREE are present by default, but can be modified or removed. The next four, UNIXFILE, SWAPFILE, TMPDEV, and ULIMIT, can be added.

DUMPDIR specifies the system dump directory. MINFREE specifies the minimum number of 512 byte blocks that should be left free on the filesystem containing DUMPDIR. If the dump, or /unix, or both will not fit in DUMPDIR without leaving MINFREE blocks free, it will not be saved.

Typically UNIXFILE is /unix and SWAPFILE is /dev/swap. TMPDEV is the pathname of a device that will be created and removed by dumpsave when the dump spans more than one swap device. The value of ULIMIT is normally unlimited, but can be set lower.

OPTIONS

−c Checks if the dump exists, but will not save or clear it. Returns 0 if the dump exists, otherwise 1.

−q Quiet. A less verbose mode.

−d *dumpdir*
Directory to which the dump file and /unix will be copied.

−s *swapfile*
The source of the dump will be *swapfile* instead of /dev/swap.

−o The memory dump is not written to the directory specified with *dumpdir* rather to standard output for further processing. This option is intended for writing the memory dump to tape.

DIAGNOSTICS

The dumpsave command exits with a status of 0 if the dump exists and was successfully saved, otherwise non-zero.

FILES

/dev/swap
/dev/tmpdumpdev

```
/etc/default/dumpsave
/unix
/var/crash
```

SEE ALSO

dumpsave(4).

System Administrator's Reference Manual

edquota(1M)

NAME

`edquota` - edit user quotas

SYNOPSIS

`edquota` [`–p` *proto_user*] *username* ...

`edquota –t`

DESCRIPTION

`edquota` is a quota editor. One or more users may be specified on the command line. For each user a temporary file is created with an ASCII representation of the current disk quotas for that user for each mounted ufs file system that has a `quotas` file, and an editor is then invoked on the file. A null entry is used if no `quotas` file exists for a file system. The quotas may then be modified, new quotas added, etc. Upon leaving the editor, `edquota` reads the temporary file and modifies the binary quota files to reflect the changes made.

The editor invoked is `vi(1)` unless the `EDITOR` environment variable specifies otherwise.

Only the superuser may edit quotas. In order for quotas to be established on a file system, the root directory of the file system must contain a file, owned by root, called `quotas`. See `quotaon(1M)` for details.

proto_user and *username* can be numeric, corresponding to the uid of a user. Unassigned uids may be specified; unassigned names may not. In this way, default quotas can be established for users who are later assigned a uid.

OPTIONS

`–p` Duplicates the quotas of the *proto_user* specified for each *username* specified. This is the normal mechanism used to initialize quotas for groups of users.

`–t` Edits the soft time limits for each file system. If the time limits are zero, the default time limits in `/usr/include/sys/fs/ufs_quota.h` are used. Time units of sec(onds), min(utes), hour(s), day(s), week(s), and month(s) are understood. Time limits are printed in the greatest possible time unit such that the value is greater than or equal to one.

FILES

`quotas`
: quota file at the file system root

`/etc/mnttab`
: table of mounted file systems

SEE ALSO

`vi(1)`, `quota(1M)`, `quotacheck(1M)`, `quotaon(1M)`, `repquota(1M)`.

System Administrator's Reference Manual

edsysadm(1M)

NAME

`edsysadm` - sysadm interface editing tool

SYNOPSIS

`edsysadm`

DESCRIPTION

`edsysadm` is an interactive tool that adds or changes either menu and task definitions in the `sysadm` interface. It can be used to make changes directly online on a specific machine or to create changes that will become part of a software package. The command creates the administration files necessary to achieve the requested changes in the interface and either places them in the appropriate place for on-line changes or saves them to be included in a software package.

`edsysadm` presents several screens, first prompting for which type of menu item you want to change, `menu` or `task`, and then for what type of action to take, `add` or `change`. When you select `add`, a blank menu or task definition (as described below) is provided for you to fill in. When you select `change`, a series of screens is presented to help identify the definition you wish to change. The final screen presented is the menu or task definition filled in with its current values, which you can then edit.

The menu definition prompts and their descriptions are:

`Menu Name`
: The name of the new menu (as it should appear in the lefthand column of the screen). This field has a maximum length of 16 alphanumeric characters.

`Menu Description`
: A description of the new menu (as it should appear in the righthand column of the screen). This field has a maximum length of 58 characters and can consist of any alphanumeric character except at sign (@), carat (^), tilde (~), back grave (`), grave ('), and double quotes (").

`Menu Location`
: The location of the menu in the menu hierarchy, expressed as a menu pathname. The pathname should begin with the main menu followed by all other menus that must be traversed (in the order they are traversed) to access this menu. Each menu name must be separated by colons. For example, the menu location for a menu entry being added to the Applications menu is `main:applications`. *Do not include the menu name in this location definition.* The complete pathname to this menu entry will be the menu location plus the menu name defined at the first prompt.

  This is a scrollable field, showing a maximum of 50 alphanumeric characters at a time.

`Menu Help File Name`
: Pathname to the item help file for this menu entry. If it resides in the directory from which you invoked `edsysadm`, you do not need to give a full pathname. If you name an item help file that does not exist, you are placed in an editor (as defined by `$EDITOR`) to create one. The new file is created in the current directory and named `Help`.

The task definition prompts and their descriptions are:

`Task Name`
: The name of the new task (as it should appear in the lefthand column of the screen). This field has a maximum length of 16 alphanumeric characters.

`Task Description`
: A description of the new task (as it should appear in the righthand column of the screen). This field has a maximum length of 58 characters and can consist of any alphanumeric character except at sign (@), carat (^), tilde (~), back grave (`), grave ('), and double quotes (").

`Task Location`
: The location of the task in the menu hierarchy, expressed as a pathname. The pathname should begin

with the main menu followed by all other menus that must be traversed (in the order they are traversed) to access this task. Each menu name must be separated by colons. For example, the task location for a task entry being added to the applications menu is `main:applications`. *Do not include the task name in this location definition*. The complete pathname to this task entry will be the task location as well as the task name defined at the first prompt.

This is a scrollable field, showing a maximum of 50 alphanumeric characters at a time.

`Task Help File Name`
Pathname to the item help file for this task entry. If it resides in the directory from which you invoked `edsysadm`, you do not need to give a full pathname. If you name an item help file that does not exist, you are placed in an editor (as defined by `$EDITOR`) to create one. The new file is created in the current directory and named `Help`.

`Task Action`
The FACE form name or executable that will be run when this task is selected. This is a scrollable field, showing a maximum of 58 alphanumeric characters at a time. This pathname can be relative to the current directory as well as absolute.

`Task Files`
Any FACE objects or other executables that support the task action listed above and might be called from within that action. *Do not include the help file name or the task action in this list*. Pathnames can be relative to the current directory as well as absolute. A dot (.) implies "all files in the current directory" and includes files in subdirectories.

This is a scrollable field, showing a maximum of 50 alphanumeric characters at a time.

Once the menu or task has been defined, screens for installing the menu or task or saving them for packaging are presented. The package creation or on-line installation is verified and you are informed upon completion.

NOTES

For package creation or modification, this command automatically creates a menu information file and a `prototype` file in the current directory (the directory from which the command is executed). The menu information file is used during package installation to modify menus in the menu structure. A `prototype` file is an installation file which gives a listing of package contents. The `prototype` file created by `edsysadm` lists the files defined under task action and gives them the special installation class of "admin". The contents of this `prototype` file must be incorporated in the package `prototype` file.

For on-line installation, `edsysadm` automatically creates a menu information file and adds or modifies the interface menu structure directly.

SEE ALSO

`delsysadm`(1M), `pkgmk`(1M), `sysadm`(1M), `prototype`(4).

System Administrator's Reference Manual

eqn(1M)

NAME

`eqn, neqn, checkeq` - typeset mathematics

SYNOPSIS

`eqn` [−d*xy*] [−p*n*] [−s*n*] [−f*n*] [−T*dev*] [−] [*file*] ...

`neqn` [−d*xy*] [−p*n*] [−s*n*] [−f*n*] [−] [*file*] ...

`checkeq` [*file*] ...

DESCRIPTION

The `eqn` and `neqn` commands are language processors to assist in describing equations. `eqn` is a preprocessor for `troff(1M)` and is intended for devices that can print `troff`'s output. `neqn` is a preprocessor for `nroff(1M)` and is intended for use with terminals.

`checkeq` reports missing or unbalanced delimiters and `.EQ`/`.EN` pairs.

If you do not specify *file*s (or if you specify − as the last argument), `eqn` reads from the standard input. A line beginning with `.EQ` marks the start of an equation; the end of an equation is marked by a line beginning with `.EN`. Neither of these lines is altered, so they may be defined in macro packages to get centering, numbering, etc. It is also possible to set two characters as "delimiters"; subsequent text between delimiters is also treated as `eqn` input.

OPTIONS

−d*xy* Sets equation delimiters set to characters *x* and *y* with the command line argument. The more common way to do this is with `delim`*xy* between `.EQ` and `.EN`. The left and right delimiters may be identical. Delimiters are turned off by `delim off` appearing in the text. All text that is neither between delimiters nor between `.EQ` and `.EN` is passed through untouched.

−f*n* Change font to *n* globally in the document. The font can also be changed globally in the body of the document by using the `gfont` directive.

−p*n* Reduce subscripts and superscripts by *n* point sizes from the previous size. In the absence of the −p option, subscripts and superscripts are reduced by 3 point sizes from the previous size.

−s*n* Set equations in point size *n* globally in the document. The point size can also be changed globally in the body of the document by using the `gsize` directive.

−T*dev*
Prepare output for device *dev*. If no −T option is present, `eqn` looks at the environment variable `TYPESETTER` to see what the intended output device is. If no such variable is found in the environment, a system-dependent default device is assumed. Not available using `neqn`.

USAGE

eqn Language

Tokens within `eqn` are separated by braces, double quotes, tildes, circumflexes, SPACE, TAB, or NEWLINE characters. Braces { } are used for grouping; generally speaking, anywhere a single character like *x* could appear, a complicated construction enclosed in braces may be used instead. Tilde (~) represents a full SPACE in the output, circumflex (^) half as much.

Subscripts and superscripts are produced with the keywords `sub` and `sup`. Thus '`x sub i`' makes x_i, '`a sub i sup 2`' produces a_i^2, and '`e sup {x sup 2 + y sup 2}`' gives $e^{x^2+y^2}$.

Fractions are made with `over`: '`a over b`' yields $\frac{a}{b}$

`sqrt` makes square roots: '`1 over down 10 sqrt {ax sup 2 +bx+c}`' results in

$$\frac{1}{\sqrt{ax^2+bx+c}}$$

Although `eqn` tries to get most things at the right place on the paper, occasionally you will need to tune the output to make it just right. In the previous example, a local motion, `down 10` was used to get more space between the square root and the line above it.

The keywords `from` and `to` introduce lower and upper limits on arbitrary things:

$$\lim_{n \to \infty} \sum_{0}^{n} x_i$$

is made with '`lim from {n-> inf } sum from 0 to n x sub i`'.

Left and right brackets, braces, etc. of the right height are made with `left` and `right`:

'`P~=~R~left [ 1^-^{1+i sup n } over alpha right ]`' produces

$$P = R \left[1 - \frac{1+i^n}{\alpha} \right]$$

The `right` clause is optional. Legal characters after `left` and `right` are braces, brackets, bars, `c` and `f` for ceiling and floor, and `""` for nothing at all (useful for a right-side-only bracket).

Vertical piles of things are made with `pile`, `lpile`, `cpile`, and `rpile`:

'`pile {a above b above c}`' produces

$$\begin{matrix} a \\ b \\ c \end{matrix}$$

There can be an arbitrary number of elements in a pile. `lpile` left-justifies, `pile` and `cpile` center, with different vertical spacing, and `rpile` right justifies.

Matrices are made with `matrix`:

```
left ( matrix {
   ccol { a sub 1 above a sub 2 }
   ccol { c sup 2 above d sup 2 }
} right )
```

produces

$$\begin{pmatrix} a_1 & c^2 \\ a_2 & d^2 \end{pmatrix}$$

In addition, there is `rcol` for a right-justified column.

Diacritical marks are made with `dot`, `dotdot`, `hat`, `tilde`, `bar`, `vec`, `dyad`, and `under`:

'`x dot = f(t) bar`' yields $\dot{x} = \overline{f(t)}$, '`y dotdot bar ~=~ n under`' yields $\underline{\bar{\ddot{y}}} = n$, and

'`x vec ~=~ y dyad`' yields $\vec{x} = \overleftrightarrow{y}$

Sizes and font can be changed with `size` *n* or `size` ±*n*, `roman`, `italic`, `bold`, and `font` *n*. Size and fonts can be changed globally in a document by `gsize` *n* and `gfont` *n*, or by the command-line arguments `-s`*n* and `-f`*n*.

Successive display arguments can be lined up. Place `mark` before the desired lineup point in the first

equation; place `lineup` at the place that is to line up vertically in subsequent equations.

Shorthands may be defined or existing keywords redefined with `define`:

`define` *thing* `%` *replacement* `%`

defines a new token called *thing* which will be replaced by *replacement* whenever it appears thereafter. The `%` may be any character that does not occur in *replacement*.

Keywords like `sum` ($\sum$), `int` ($\int$), `inf` (∞), and shorthands like `>=` ($\geq$), `->` ($\rightarrow$), and `!=` ($\neq$) are recognized. Greek letters are spelled out in the desired case, as in `alpha` or `GAMMA`. Mathematical words like `sin`, `cos`, and `log` are made Roman automatically. `troff(1M)` four-character escapes like `\(dd` ([Dag]) can be used anywhere. Strings enclosed in double quotes `"..."` are passed through untouched; this permits keywords to be entered as text, and can be used to communicate with `troff` when all else fails.

EXAMPLES

`eqn` [*options*] *files* | `troff` [*options*] | [*phototypesetter*]

`neqn` [*options*] *file* ... | `nroff` [*options*] | [*printer*]

NOTES

To embolden digits, parentheses, etc. it is necessary to quote them, as in `bold "12.3"`. When you use `eqn` with the `mm` macro package, displayed equations must appear only inside displays.

SEE ALSO

`neqn(1M)`, `nroff(1M)`, `troff(1M)`, `eqnchar(5)`.

REFERENCES

B. W. Kernighan, L. L. Cherry, *Typesetting Mathematics — User's Guide* (2nd Ed.)

D. Dougherty, T. O'Reilly, *UNIX TEXT PROCESSING*, Hayden Books, Indianapolis, Indiana 46268 USA, ISBN 0-672-46291-5

System Administrator's Reference Manual

errlogdo(1M)

NAME

errlogdo - DO command for TMS (Threshold Monitoring System)

SYNOPSIS

errlogdo –c *class_function* [...]

errlogdo –c ~typecreate –o *object_name* [–l|–d *object_name*]

errlogdo –c ~typeremove –o *object_name*

errlogdo –c errlog~cp –t *file* –e *object_name*

errlogdo –c errlog~cp –t *file* –i *object_name* [–p *protection*]

errlogdo –c ~linkcheck –o *object_name*

errlogdo –H

errlogdo –h

DESCRIPTION

The errlogdo command can be used to have various operations or functions executed within TMS (threshold monitoring for SCSI devices). Some functions are closely linked to object classes while others have no unique affinity to an object class, rather affect TMS as a whole (e.g. activities of the modified vendor limit files).

The errlogdo command is the DO command for TMS that can be used to execute various operations and functions within TMS, which do not uniquely affect object classes, which affect several object classes at the same time or which are not one of the classic functions (create, modify and delete data records) within an object class.

Note:

Because the functions are not the same, the function calls have quite different call parameters. No plausibility, admissibility or validity check is performed for the data transferred in the calls.

TMS is a component of the kernel and offers its own FMLI user interface (package name SIerrlog).

Vendor limit files always either define a hard or soft limit in the case of sense codes in SCSI devices. If these limits are exceeded, the device is classified as faulty or no longer usable.

If no function class has been specified, the $CLASS environment variable is evaluated.

OPTIONS

–c *class-function*

This option determines the function to be executed. There are special functions for the three classes, i.e. disk, tape and mem. These are marked in the syntax overview above by the "~" variable.

activate
: Activates changes in vendor limit files

disktypecreate
: Creates a vendor limit file for SCSI disks

tapetypecreate
: Creates a vendor limit file for SCSI tapes

memtypecreate
: Creates a vendor limit file for main memory and CPU cache

disktyperemove
: Deletes a vendor limit file for SCSI disks

tapetyperemove
: Deletes a vendor limit file for SCSI tapes

`memtyperemove`
Deletes a vendor limit file for main memory and CPU cache

`errlogdiskcp`
Exports a vendor limit file to a temporary file or integrates it from there

`errlogtapecp`
Exports a vendor limit file to a temporary file or integrates it from there

`errlogmemcp`
Exports a vendor limit file to a temporary file or integrates it from there

`disklinkcheck`
Checks whether the name of a vendor limit file is a link

`tapelinkcheck`
Checks whether the name of a vendor limit file is a link

`memlinkcheck`
Checks whether the name of a vendor limit file is a link

`-o` *object_name*
Name of the vendor limit file to be checked, created or deleted.

`-t` *file*
Name of the temporary file to which a vendor limit file is to be copied or from which a vendor limit file is to be integrated, i.e. from which it is to be copied to the directory for vendor limit files on the relevant system.

`-i` *object_name*
Name of the vendor limit file to be integrated.

`-e` *object_name*
Name of the vendor limit file to be extracted.

Note:

The command outputs the current protection for the extracted file to standard output:

`PROTECTION=protected`

or

`PROTECTION=changeable`

`-p` *protection*
The `-p` option can be used to predefine how the vendor limit file to be integrated is to be protected on the destination system. A new file is protected by default on the destination system, i.e. assigned the label `protected`. It then cannot be edited anymore. If the vendor limit file is to remain editable following copying, the protection label `changeable` must be preset.

`-l` *object_name*
The new file being created is to act as a link to the file specified with the `-l` option.

`-d` *object_name*
The new file being created is to act as a copy of the file specified with the `-d` option.

`-H` The list of known `errlogdo` function classes is output (one name per line).

`-h` Usage for this command.

DIAGNOSTICS

The return value is 0 is no error is detected. If the `errlogdo` command discovers an error situation, this is reported to standard error and the return value is not 0.

`1` `"Illegal/less options!"`

Bad call: Incorrect command options or option values were used.

`2` `"Logging V3.0 not installed / not found!"`

The Logging V3.0 home directory cannot be established from the package information. This indicates

that Logging V3.0 was not installed.

3 "Command update_SCSI not found?!"

The update_SCSI access command cannot be found.

4 "You are not authorised to restart monitoring of SCSI devices!"
"You are not authorised to administrate Logging V3.0!"

You are not authorized as a system administrator for Logging V3.0 or to use threshold monitoring. You are therefore not permitted to either create or delete vendor limit files.

5 "Unknown class function "........."!"

The requested function does not exist or is not known to the current errlogdo command.

6 "Vendor limit file already exists!"

The required call data was not transferred.

7 "Illegal/no data ?!"

The required call data was not transferred.

8 "Vendor limit file doesn't exist!"

The required call data was not transferred.

10 "Sorry, not implemented!"

Resetting of the counter states in TMS has not been implemented, because the errlogd command does not (yet) offer this functionality.

13 "Table cannot be deleted - Number of links: *nn*"

A threshold table (or vendor limit file) cannot be deleted, as long as there are still links to the file. You must first remove all links to this file before the file itself can be deleted.

14 "Vendor limit file is protected!"

Protected vendor limit files can only be deleted if the file protection has first been removed manually.

15 "Activate vendor limit files results in error!"

Activation of vendor limit files returns errors.

NOTES

If requested data is not available, this is not reported as an error. The output is blank in this case and the return value is 0. The newly created vendor limit files only contain comments but no data records.

FILES

/etc/default/diskinfo/errlogd/*
/etc/default/tapeinfo/errlogd/*
/etc/default/meminfo/errlogd/*

SEE ALSO

errlogget(1M), errlogreplace(1M), errlogd(8), errord(8).

System Administrator's Reference Manual

errlogget(1M)

NAME

`errlogget - GET` command for TMS (Threshold Monitoring System)

SYNOPSIS

`errlogget` [`–c` *object_class*] [`–a` *attribute_value...*] [`–s` "*selection*"...] [`–V` *sep*] [`–f` *num*] [`–t` *num*]

`errlogget` [`–c` *object_class*] [`–H`]

`errlogget –h`

DESCRIPTION

The `errlogget` command can be used to output all threshold monitoring data for SCSI devices (TMS). This includes both the threshold definitions and the current counter states (not yet implemented).

The `errlogget` command is the `GET` command for threshold monitoring, which can be used to fetch the unformatted data for all object classes. The information is output on a line by line basis for each data record. The attribute fields are separated by a freely definable separator. If the separator occurs in the data, it is masked with "\". The attributes can be numbered and ordered on the basis of a predefined list of attribute names. Specific data records can be selected by predefining selection criteria.

Notes:

If the attribute columns requested are unknown, no error message is issued rather the columns remain blank. This means that the same set of attribute columns can be requested on systems running any releases of Logging V3.0 without error messages being issued.

If no object class has been specified, the `$CLASS` environment variable is evaluated.

OPTIONS

`–c` *object_class*

This option determines the object class from which data is to be read. The object class names known at present are as follows (see `–H` option):

`errlogdisktypes`
: List of available threshold tables for disks with protection

`errlogdisks`
: Threshold values for the sense codes of all disks

`errlogtapetypes`
: List of available threshold tables for tapes with protection

`errlogtapes`
: Threshold values for the sense codes of all tapes

`errlogmemtypes`
: List of available threshold tables for CPU cache and main memory with protection

`errlogmem`
: Threshold values for the sense codes of all CPU caches and main memory

`errlogcounters`
: Counter states (not yet implemented)

`–a` *attribute_value...*

The *attribute_value* argument is used to define a selection of attribute columns as well as the order in which they will be output.

The information is provided in the form of a list of attribute names, separated by the `@` character (a fixed separator, which does not occur in the attribute name). The known attribute names can be output with the `errlogget –c` *object_class* `–H` command.

Example: `-a diskType@senseCodeK@senseCodeC@senseCodeQ`

Returns the disk type and the sense codes (for which threshold values are defined).

`-s "`*selection*`"...`

A list of expressions in the form *AttributenameOpValue* can be specified as a selection criterion for data records, where *Attributename* is one of the known attribute names (see `-H` option), *Op* is one of the operators ==, !=, ~=, <, <=, >, >= and *Value* is the requested attribute value. The selection criteria are linked in list form with "AND".

The operators can be surrounded by any number of blanks and tabs.

The entire list must be combined to form a single argument with "...".

Example: `-s "diskType ~= QUANTUM AND protection == protected"`

Returns the names of all disk types whose names begin with QUANTUM and which are protected.

`-V` *sep*

Defines a specific separator *sep*. Only a single character is permitted. If this character occurs in the readable ASCII data, it is masked with "\". The default separator is CAN (octal 030), a non-printable character, which cannot occur in the readable ASCII data.

`-f` *num*

Predefines the line number where data output is to begin.

Example: `-f 201`

The 201st data record is output first.

`-t` *num*

Predefines the line number where data output is to end.

Example: `-t 100`

The first 100 data records at most are output.

`-H` If the option appears on its own, the list of object classes known to `errlogget` is output (one name per line). If an object class name is specified in addition with the `-c` option, the list of attribute names for this object class is output (one line with the attribute names separated by the `@` character).

`-h` Usage for this command.

Note:

The `-f` and `-t` options allow the data records to be read out in sections. Considerably more data records may have to be read out internally in order to output the required section.

DIAGNOSTICS

The return value is 0 if no error is detected. If the `errlogget` command discovers an error situation, this is reported to standard error output and the return value is not 0.

`1` "`Illegal/less options!`"

Bad call: Incorrect command options or option values were used.

`2` "`OA&M extensions not installed / not found!`"

The OA&M home directory cannot be found. This means that the access commands it has stored for threshold tables are not available.

`3` "`SIerrlog not installed / not found!`"

The `SIerrlog` package cannot be found.

`4` "`You are not authorised to administrate Logging V3.0!`"

You are not authorized as a system administrator for Logging V3.0; protected data is therefore not available to you.

`5` "`Unknown object class!`"

The specified object class does not exist or is not known to the current `errlogget` command.

`10` `"Sorry, not implemented!"`

The output of counter states is not yet implemented because there is no command for accessing the current states of the monitored devices.

NOTES

The speed at which the data can be established depends on the requested attribute columns and the specified selection criteria. If all of the known attribute columns are requested, no filtering generally takes place. Additional filtering with `awk` is implied for all object classes where specific attribute columns are requested and selection criteria are predefined.

The system name (node) is provided as an attribute field (second column) for all object classes.

FILES

`/etc/default/diskinfo/errlogd/*`
`/etc/default/tapeinfo/errlogd/*`
`/etc/default/meminfo/errlogd/*`

SEE ALSO

`errlogdo`(1M), `errlogreplace`(1M), `errlogd`(8), `errord`(8).

System Administrator's Reference Manual

errlogreplace(1M)

NAME

errlogreplace - SET command for TMS (Threshold Monitoring System)

SYNOPSIS

errlogreplace [–c *object_class*] [–o *object_name*] [–r "*attribute_value*"...] [–V *sep*]

errlogreplace [–c *object_class*] –o *object_name*

errlogreplace –h

DESCRIPTION

The errlogreplace command can be used to change, delete or insert any data records in vendor limit files. The counter states for threshold monitoring are altered by external events (incoming messages etc.) and cannot be modified by a command call.

The errlogreplace command is the SET command for threshold monitoring, which can be used to manipulate the data record of any object, to create a new data record and thus a new object and to delete objects.

Notes:

If no object class has been specified, the $CLASS environment variable is evaluated.

If no attribute values are specified (no –r option), the specified object is deleted.

If no object name has been specified, the new object name is taken from the new attribute value settings.

If no object of that name already exists, it will be created by appending the new data record to the relevant file. No error message is issued if a non-existent record is to be replaced, rather a new record is created.

OPTIONS

–c *object_class*

This option determines the object class where changes are to be made. The object class names known at present are as follows [see –H option in errlogget(1M)]:

errlogdisks
: All sense code entries from all vendor limit files for disks

errlogtapes
: All sense code entries from all vendor limit files for tapes

errlogmem
: All sense code entries from all vendor limit files for CPU cache and main memory

–o *object_name*

The *object_name* argument determines the data record to be modified, deleted or created or also the object.

In each object class, a different attribute field acts as a unique identifier for an object. In principle, it would also be possible for only a combination of attribute fields to uniquely identify an object. In extreme cases, only the entire data record with all attribute fields acts as a unique identifier. The objects are identified by the following attribute fields in the object classes:

errlogdisks
: diskType:senseCodeK:senseCodeC:senseCodeQ

errlogtapes
: tapeType:senseCodeK:senseCodeC:senseCodeQ

errlogmem
: memType:senseCodeK:senseCodeC:senseCodeQ

–r "*attribute_value*"...

The new attribute values are specified as a list of expressions in the form *attribute_name*=*value*, where *attribute_name* is one of the known attribute names [see –H option in errlogget(1M)] and *value* is the attribute value. Several such expressions are separated by a freely definable separator.

The "=" operator can be surrounded by any number of blanks and tabs. The entire list must be combined to form a single argument with "...".

Several –r options can be used.

Example:

```
–r "diskType=SEAGATE_ST34573LC @ senseCodeC=01 @ senseCodeK=01 @
senseCodeQ=01"
```

–V *sep*
: Defines a specific separator. Only a single character is permitted. If this character occurs in the readable ASCII data, it must be masked with "\" in both input and output. The default separator is @.

–h
: Usage for this command.

DIAGNOSTICS

The return value is 0 if no error is detected. If the errlogreplace command discovers an error situation, this is reported to standard error output and the return value is not 0.

1 "Illegal/less options!"
: Bad call: Incorrect command options or option values were used.

2 "SIerrlog not installed / not found!"
: The SIerrlog package cannot be found.

3 "OA&M extensions not installed / not found!"
: The home directory for access commands (e.g. update_SCSI) cannot be established.

4 "You are not authorised to administrate Logging V3.0!"
: You are not authorized as a system administrator for Logging V3.0; protected data is therefore not available to you.

5 "Unknown object class ".....!"
: The specified object class does not exist or is not known to the current errlogreplace command.

9 "Missing device type!"
: No device type (diskType, tapeType, memType etc.) was specified.

10 "Illegal/missing sense code!"
: No sense code or an invalid sense code was specified.

11 "No device type in new record!"
: The new data record does not contain a device type.

12 "Device type (".....") differs from new record (".....")!"
: The old and the new data records have different device types. However, it is not permitted to change the object name. The old data record can be deleted in two steps and the new data record entered.

13 "Vendor limit file is protected!"
: The vendor limit file is protected and cannot be changed without first removing its protection.

14 "Cannot make changes in vendor limit file of this type!"
: The data record can neither be changed nor reentered because either the file does not exist or a link is in question. Therefore, either the file must be created first or the original file must be changed instead of the link.

16 "No rename of sense code C (please delete and insert)!"
: It is not permitted to change the object name. The data record can only be deleted and then a new data record entered with the new sense code.

17 "Contradiction in sense codes (please delete first)!"

You are misusing the "*" operator: If a threshold value is defined for a class of sense codes, a different threshold value cannot be agreed for a single sense code from this class.

NOTES

The new attribute values are not entered in existing data records, rather all objects with the specified object name are deleted and a new data record is inserted with the new attribute values. This guarantees the uniqueness of the data records and thus the objects following conclusion of the write process. An existing object is thus replaced in this way by a new object with the same name (identifier) but with different attribute values. The position of the object in the file will generally change as a result of this.

The data records are entered with a fixed layout in the vendor limit files. This layout may differ from the existing file layout.

If individual attribute values are changed specifically, the data record must be read, the relevant attribute fields must be modified and the data record must be fully restored.

The attribute values are not checked for admissibility, plausibility or completeness.

FILES

/etc/default/diskinfo/errlogd/*
/etc/default/tapeinfo/errlogd/*
/etc/default/meminfo/errlogd/*

SEE ALSO

errlogdo(1M), errlogget(1M), errlogd(8), errord(8).

System Administrator's Reference Manual

errsum(1M)

NAME

`errsum -` summarize system log files

SYNOPSIS

`/usr/bin/errsum –f` *file* [`–d` *database*]

DESCRIPTION

`errsum` summarizes the header section of an error log file. `errsum` is mostly concerned with the "type", "severity", "prog" and "entity" fields. For a complete description of the header fields, see `errord`(8) . For each error log file, errsum prints the following:

1. summary of the type and severity for each entity
2. summary of the type and severity for each prog
3. total summary for type and severity
4. total log entries, first and last log date

OPTIONS

`–f` The name of the error log *file* to summarize

`–d` The name of the *database* where the error log file resides. This option is only needed if the database has been moved from its default location of `/var/adm/error/errlog.db`.

FILES

`/var/adm/error/vsm_errfile.current`
current log file

`/var/adm/error/errlog.db`
default log database

SEE ALSO

`erruser`(1M) , `error`(7) , `errord`(8) .

System Administrator's Reference Manual

erruser(1M)

NAME

`erruser` - browse through system error log files

SYNOPSIS

`/usr/bin/erruser` `–f` *file* `–p` *pattern* [`–C` *criteria_file*]

DESCRIPTION

`erruser` allows a user to browse through the vsm error log files that reside in the `/var/adm/error` directory. Since these files are in binary format, they are unreadable and must be interpreted.

OPTIONS

`–f` The name of the error log *file* to browse through.

`–p` device *pattern*. Use this to browse through log entries that were made by a specific device logger. For example to browse through only the entries that were logged by the `ios` disk driver, specify "pyr_iosdisk". The pattern can be any regular expression that conforms to `gmatch`(3G) . More examples are given below in the **EXAMPLES** section.

`–C` full pathname of *criteria_file*. A criteria file is used to carry out extensive filtering of log entries. Here are some rules regarding the criteria.

The format of a criteria file is:

identifier operator value [*logical identifier operator value* ...] `$`

Legal *operators* are:

`EQ` – Equal to

`NEQ` – Not equal to

`LT` – Less than

`LTE` – Less than or equal to

`GT` – Greater than

`GTE` – Greater than or equal to

Legal *identifiers* and their range of field operators:

`log date`
`EQ, LT, LTE, GT, GTE`

`error type`
`EQ, NEQ`

`severity`
`EQ, NEQ`

`device name`
`EQ, NEQ`

Legal *values* are:

`log date`
"*Month day year*"

`error type`
`ET_NONE, ET_GOTS, ET_STOP, ET_SOFT, ET_HARD, ET_MEDIA`

`severity`
`ES_NONE, ES_NOTE, ES_WARN, ES_ERROR, ES_PANIC, ES_CONT, ES_COR, ES_UNCOR`

`device name`
Any regular expression that conforms to `gmatch`(3G) .

logical operators are:

`AND` And

`OR` Or

NOTES

Any field identifier or field value that is more than one word in length must be enclosed in quotes. Do not forget the `$` sign at the end of line in the criteria file.

EXAMPLES

Specifying a device pattern:

To print all log entries that were logged by a SCSI driver:

`erruser –f` *file* `–p pyr_ios*`

To print all log entries in a file:

`erruser –f` *file* `–p pyr_*`

Criteria format:

To print the log entries that have a severity of `ES_ERROR`:

```
severity EQ ES_ERROR $
```

To print all the log entries that have a severity of `ES_ERROR` and an error type of `ET_HARD`:

```
severity EQ ES_ERROR AND "error type" EQ ET_HARD $
```

To print all the log entries that were logged between April 1st, 1991 and June 12th, 1991:

```
"log date" GTE "Apr 1 1991" AND "log date" LTE "Jun 12 1991" $
```

FILES

`/var/adm/error/vsm_errfile.current`
current vsm error log file

`/var/adm/error/errlog.db`
default vsm error log database

SEE ALSO

`errsum`(1M), `error`(7), `errord`(8).

fastboot(1M) - fwtmp(1M)

System Administrator's Reference Manual

fastboot(1M)
BSD Compatibility Package

NAME

`fastboot, fasthalt -` reboot/halt the system without checkingthe disks

SYNOPSIS

`/usr/ucb/fastboot` [*boot-options*]

`/usr/ucb/fasthalt` [*halt-options*]

DESCRIPTION

`fastboot` and `fasthalt` are shell scripts that invoke `reboot`(1M) and `halt`(1M) with the proper arguments.

These commands are provided for compatibility only.

FILES

`/etc/rc`

SEE ALSO

`fsck`(1M), `halt`(1M), `init`(1M), `rc0`(1M), `rc2`(1M), `rc6`(1M), `reboot`(1M).

System Administrator's Reference Manual

fdetach(1M)

NAME

`fdetach -` detach a name from a STREAMS-based file descriptor

SYNOPSIS

`fdetach` *path*

DESCRIPTION

The `fdetach` command detaches a STREAMS-based file descriptor from a name in the file system. *path* is the path name of the object in the file system name space, which was previously attached [see `fattach`(3C)]. The user must be the owner of the file or a user with the appropriate privileges. All subsequent operations on *path* will operate on the file system node and not on the STREAMS file. The permissions and status of the node are restored to the state the node was in before the STREAMS file was attached to it.

SEE ALSO

`fattach`(3C) , `fdetach`(3C) , `streamio`(7) .

Programmer's Guide: STREAMS

System Administrator's Reference Manual

ff(1M)

NAME

`ff(generic)` - list file names and statistics for a file system

SYNOPSIS

`ff` [`-F` *FSType*] [`-VIlsu`] [`-p`*prefix*] [`-a`*n*] [`-m`*n*] [`-c`*n*] [`-n`*file*] [`-i`*i-node-list*] [`-o` *specific_options*] *special* ...

DESCRIPTION

`ff` reads the files and directories of the *special* file. *special* must be an unmounted file system. I-node data is saved for files which match the selection criteria which is either the *i-node* number and/or *i-node* age. Output consists of the path name for each saved i-node, plus other file information requested using the print *options* below. Output fields are positional. The output is produced in i-node order; fields are separated by tabs. The default line produced by `ff` is:

path-name i-number

The path name is preceded by a dot unless the `-p` option is specified.

The maximum information the command will provide is:

path-name i-number size uid

There are two types of option: *generic_options* and *specific_options*. *generic_options* indicate options that are supported not only by the generic but also by the *FST*-specific module of the command (if that command exists). *specific_options* indicate suboptions that are supported only by the *FST*-specific module of the command. They are specified in a comma-separated list and/or as keyword-attribute pairs for interpretation by the *FST*-specific module of the command. These suboptions are specified via the `-o` option.

The argument *n* in the *option* descriptions that follow is used as a decimal integer (optionally signed), where `+`*n* means more than *n*, `-`*n* means less than *n*, and *n* means exactly *n*. A day is defined as a 24 hour period.

OPTIONS

The *generic_options* are:

`-F` Specifies the *FSType* on which to operate. The *FSType* should either be specified here or be determinable from `/etc/vfstab` by matching the *special* with an entry in the table.

`-V` Echoes the complete command line, but does not execute the command. The command line is generated by using the options and arguments provided by the user and adding to them information derived from `/etc/vfstab`. This option should be used to verify and validate the command line. When used with the *FSType* option, it passes the `-V` to the *FSType*-specific module for interpretation.

`-I` Prevents the i-node number being printed after each path name.

`-l` Generates a supplementary list of all path names for multiply-linked files.

`-p`*prefix*
Adds the specified *prefix* to each generated path name. The default is " . " (dot).

`-s` Prints the file size, in bytes, after each path name.

`-u` Prints the owner's login name after each path name.

`-a`*n* Select if the i-node has been accessed in *n* days.

`-m`*n* Select if the i-node has been modified in *n* days.

`-c`*n* Select if the i-node has been changed in *n* days.

`-n`*file*
Select if the i-node has been modified more recently than the argument *file*.

`-i`*i-node-list*

Generates names for only those i-nodes specified in *i-node-list*. *i-node-list* is a list of numbers separated by commas and without spaces.

`-o` Specifies FSType-specific options.

NOTES

If the `-l` option is not specified, only a single path name out of all possible ones is generated for a multiply-linked i-node. If `-l` is specified, all possible names for every linked file on the file system are included in the output. However, no selection criteria apply to the names generated.

This command may not be supported for all FSTypes.

FILES

`/etc/vfstab`
List of default parameters for each file system

SEE ALSO

`find`(1), `ncheck`(1M), `vfstab`(4).

Manual pages for the FSType-specific modules of `ff`.

System Administrator's Reference Manual

ff(1M-ufs)

NAME

`ff(ufs)` - list file names and statistics for a ufs file system

SYNOPSIS

`ff [ -F ufs ] [` *generic_options* `] [ -o a,m,s ]` *special* ...

DESCRIPTION

`ff` reads the i-list and directories of the *special* file, assuming it is an unmounted `ufs` file system. I-node data is saved for files which match the selection criteria. Output consists of the path name for each saved i-node, plus other file information requested using the *generic_options*. For more information, see the man page for the generic `ff` command.

OPTIONS

`-F ufs`
: Specifies the `ufs` FSType.

generic_options
: Options supported by the generic `ff` command.

`-o`
: Specifies `ufs` file system specific options. The specific options available are:

  `a` Allows the "`.`" and "`..`" directory entries to be printed.

  `m` Prints mode information.

  `s` Only prints special files and files with set-user-ID mode.

SEE ALSO

`find`(1) , generic `ff`(1M) , `ncheck`(1M) .

System Administrator's Reference Manual

ff(1M-vxfs)
VxFS

NAME

`ff(vxfs) -` list file names and inode information for a VeritasFile System (VxFS)

SYNOPSIS

`ff [ –F vxfs ] [` *generic_options* `] [ –o s ]` *special ...*

DESCRIPTION

This manual page is an addendum to the `ff(1M)` manual page. It describes the Veritas File System (VxFS) specific features of the `ff` command.

`ff` reads the inode list and directories of the *special* file, assuming it is a Veritas file system. Inode data is saved for files that match all the selection criteria. Output consists of the path name for each saved inode, plus other file information requested using the generic options.

OPTIONS

`–F vxfs`
Specifies the Veritas file system type (*FSType*).

generic_options
Options supported by the generic `ff` command.

`–o` Specifies Veritas file system specific options. The following option is available:

`s` Only prints special files and files with set-user-ID mode.

SEE ALSO

`find(1)`, `ff(1M)`, `ncheck(1M)`, VxFS-specific `fs`(4).

System Administrator's Reference Manual

flformat(1M)

NAME

`flformat` - format floppy disk

SYNOPSIS

`flformat` [*options*] *special_file*

DESCRIPTION

`flformat` is used to format floppy disks. `flformat` is called from the command line.

`flformat` can only format floppies for Reliant UNIX systems. Floppy disks for the MS-DOS operating system must be formatted with the `dosformat(1)` command.

Defective floppy disks cannot be used under Reliant UNIX .

ED floppy disks can be formatted on ED, HD and DD. HD floppy disks can be formatted on HD and DD. DD floppy disks can only be formatted on DD. If an autodetector device is used, the floppy disk is formatted according to its physical format.

Note:

Not all systems (non-SNI manufacturers) can read floppy disks that have been formatted incorrectly. It is possible, for example, that ED floppy disks formatted on HD cannot be recognized (because of missing floppy disk indicators).

OPTIONS

`-f` *number*
: (first) Specifies the track at which formatting is to start.

`-l` *number*
: (last) Specifies the track at which formatting is to end.

`-i` *number*
: Interleave factor to be used in formatting the floppy disk. The default is 2.

`-V`
: (verify) Select this option to indicate that a read/write verification check is to be performed on a number of randomly selected sectors.

`-E`
: (exhaustive verify) Select this option to indicate that a read/write verification check is to be performed on all the sectors of the disk.

special_file
: Specifies the storage medium to be formatted.

The following list indicates the main special files involved and the formats they support:

Special file	Supported floppy disk format
`/dev/ios?/rsfdisk00?f3dt`	3 ½", 9 sectors/track, 720 KB
`/dev/ios?/rsfdisk00?f3ht`	3 ½", 18 sectors/track, 1.44 MB
`/dev/ios?/rsfdisk00?f3et`	3 ½", 36 sectors/track, 2.88 MB
`/dev/ios?/rsfdisk00?ft`	3 ½", format depends on the physical properties of the floppy disk
`RM400:`	
`/dev/at/flp/rfx3dt`	3 ½", 9 sectors/track, 720 KB
`/dev/at/flp/rfx3ht`	3 ½", 18 sectors/track, 1.44 MB

EXAMPLES

Using `flformat` to prepare a 3 ½" floppy disk for use, you place the disk in the floppy disk drive and enter:

```
$ flformat /dev/ios0/rsfdisk000f3ht
```

While formatting the disk, `flformat` shows you how many tracks it has processed so far. A dot is displayed on the screen for every ten tracks.

FILES

`/dev/ios?/sfdisk*`[RM600 only]
`/dev/ios?/rsfdisk*`
`/dev/at/flp/fx3*`
`/dev/at/flp/rfx3*`

SEE ALSO

`dosformat(1)`, `sfdisk(7)`.

System Administrator's Reference Manual

fromsmtp(1M)

NAME

`fromsmtp` - receive RFC822 mail from SMTP

SYNOPSIS

`/usr/lib/mail/surrcmd/fromsmtp` [`-d`] [`-h` *host*] [`-s` *sender*] *to* ...

DESCRIPTION

`fromsmtp` reads an RFC822 message from its standard input, does some conversion of the message to make it acceptable to Reliant UNIX system mail, and pipes the result to `rmail`. The *to* arguments are passed as arguments to `rmail`. `fromsmtp` is normally invoked by `smtpd` to deliver incoming mail messages.

The `-d` option may be used for debugging `fromsmtp`. It will cause the command line for `rmail` to be echoed to standard output, as well as the results of the message (after conversion). The message will not be given to `rmail` when this option is used.

The `-h` *host* option may be used to prepend a host or network name to the front of the sender path in the `From` line at the beginning of the message. This is useful if you need to identify which of several possible networks a message was received from (for possible use in replying).

The `-s` *sender* option is used to give a default sender name, in case `fromsmtp` cannot determine the name of the sender from the message it reads. If this option is not used, the default sender name `unknown` will be used.

FILES

`/usr/bin/rmail`
: where converted mail is piped to

SEE ALSO

`rmail`(1), `smtpd`(1M).

RFC822 – Standard for the Format of ARPA Internet Text Messages.

System Administrator's Reference Manual

fsadm(1M)
VxFS

NAME

`fsadm -` resize or reorganize a Veritas File System (VxFS)

SYNOPSIS

`fsadm [–F vxfs ][–b` *new_size* `][–r` *raw_dev* `]` *mount_point*

`fsadm [–F vxfs ][–E ]` *mount_point*

`fsadm [–F vxfs ][–D ]` *mount_point*

`fsadm [–F vxfs ][–e ][–E ][–s ][–v ][–l` *large_size* `][–a` *days* `][–t` *time* `][–p` *passes* `][–r` *raw_dev* `]` *mount_point*

`fsadm [–F vxfs ][–d ][–D ][–s ][–v ][–a` *days* `][–t` *time* `][–p` *passes* `][–r` *raw_dev* `]` *mount_point*

DESCRIPTION

The `fsadm` utility is designed to perform file system administrative operations. The current version supports file system resizing, extent reorganization, and directory reorganization. The `fsadm` utility operates on file systems mounted for read/write access. Only the superuser can resize or reorganize file systems.

OPTIONS

`–F vxfs`
: Specifies a Veritas file system type (*FSType*).

`–b` *new_size*
: Resizes the file system to *new_size* sectors.

`–D`
: Reports on directory fragmentation. If specified in conjunction with the `-d` option, the fragmentation report is produced both before and after the directory reorganization.

`–E`
: Reports on extent fragmentation. If specified in conjunction with the `-e` option, the fragmentation report is produced both before and after the extent reorganization.

`–d`
: Reorganizes directories. Directory entries are reordered to place subdirectory entries first, then all other entries in order of time since last access. The directory is also compacted to remove free space.

`–e`
: Extent reorganization. Attempt to minimize fragmentation. Aged files are moved to the end of the allocation units to produce free space. Other files are reorganized to have the minimum number of extents possible.

`–a` *days*
: Consider files not accessed within the specified number of *days* as aged files. The default is 14 days. Aged files are moved to the end of the directory by the directory sort, and reorganized differently by the extent reorganization.

`–l` *large_size*
: Large file size in file system blocks. Indicates size of files to be considered as large files. The value must be between 8 and 2048 blocks. The default is 64 blocks.

`–p` *passes*
: Maximum number of *passes* to run. The default is five passes. Reorganizations are processed until complete or the specified number of passes have been run.

`–r` *raw_dev*
: Path name of raw device to use. Should be used when the mount point or `fsck` device for the mount point is not specified in `/etc/vfstab`.

`–t` *time*
: Maximum time to run. Reorganizations are processed until complete, or the time limit has expired. The

time is specified in seconds.

–s Prints a summary of activity at the end of each pass.

–v Verbose. Report reorganization activity.

The –b, –D, –E, –d, and –e options determine what function will be performed. At least one of these options must be selected. The –b option cannot be specified if any of the other options are specified. If both –e and –d are specified, the fsadm utility will do the directory reorganization first. It will do the extent reorganization after the directory reorganization has been completed.

For optimal performance, the kernel extent allocator must be able to find large extents when it wants them. To maintain the file system performance levels, the fsadm utility should be run periodically against all Veritas file systems to reduce fragmentation. The fsadm utility should be run somewhere between once a day and once a month against each file system. The frequency depends on file system usage and activity patterns, and the importance of performance. The –v option can be used to examine the amount of work performed by fsadm. The frequency of reorganization can be adjusted based on the rate of file system fragmentation.

There are two options that are available to control the amount of work done by the fsadm utility. The –t option is used to specify a maximum length of time to run. The –p option is used to specify a maximum number of passes to run. If both are specified, the utility exits if either of the terminating conditions is reached. By default, the fsadm utility will run two passes. If both the –e and –d options have been specified, the utility will run all the extent reorganization passes before any directory reorganization passes.

The fsadm utility uses the .fsadm file in the lost+found directory as a lock file. When fsadm is invoked, it opens the lost+found/.fsadm file in the root of the file system specified by *mount_point*. If the file does not exist, it is created. The fcntl(2) system call is used to obtain a write lock on the file. If the write lock fails, fsadm will fail because another invocation of fsadm is running against the file system. The fsadm utility will report the process ID of the process holding the write lock on the .fsadm file.

File System Resizing

If the –b option is specified, the fsadm utility will resize the file system whose mount point is *mount_point*. If *new_size* is larger than the current size of the file system, the file system will be expanded to *new_size* sectors. Similarly, if *new_size* is smaller than the current size of the file system, an attempt will be made to shrink the file system to *new_size* sectors.

Reducing the size of a file system can fail if there are file system resources currently in use within the sectors to be removed from the file system. In this case, a reorganization may help free those busy resources and allow a subsequent reduction in the file system size.

Directory Reorganization

If the –d option is specified, the fsadm utility will reorganize the directories on the file system whose mount point is *mount_point*. Directories are reorganized in two ways, compression and sorting.

For compression, the entries in the directory are moved to the front of the directory, and the free space is grouped at the end of the directory. If there are no entries in the last block of the directory, the block is released and the directory size is lowered. If the directory entries are small enough, the directory will be placed in the inode immediate data area.

The entries in a directory are also sorted to improve path name lookup performance. Entries are sorted based on the last access time of the entry. The –a option is used to specify a time interval; seven days is the default if –a is not specified. The time interval is broken up into 128 buckets, and all times within the same bucket are considered equal. All access times older than the time interval are considered equal, and those entries are placed last. Subdirectory entries are placed at the front of the directory and symbolic links are placed after subdirectories, with the most recently accessed files.

The directory reorganization runs in passes across the entire file system. On each pass, all eligible directories have a single pass of a bubble sort performed on them. When one pass through each inode has been completed, the fsadm utility will go back to the beginning of the file system and start another pass. The first pass does the first pass of a bubble sort on all directories containing one or more blocks. The second pass does the second pass of a bubble sort on all directories having two or more blocks. Each succeeding pass will

be limited to larger and larger directories. After three passes, all directories of three blocks or less will have been completely sorted. After four passes, all directories of four blocks or less will have been completely sorted, and so on.

Extent Reorganization

If the `-e` option is specified, the `fsadm` utility will reorganize the data extents on the file system whose mount point is *mount_point*. The primary goal of extent reorganization is defragmentation of the file system.

To reduce fragmentation, the reorganization tries to place all small files in one contiguous extent. The `-l` option is used to specify the size of a file that is considered large. The default is 64 blocks. The reorganization also tries to group large files into large extents of at least 64 blocks. In addition to reducing fragmentation, these reorganizations can improve performance. Small files can be read or written in one I/O operation. Large files can approach raw disk performance for sequential I/O operations.

The extent reorganization also tries to improve locality of reference on the file system. Files are moved into the same allocation unit as their inode. Within the allocation unit, small files and directories are migrated to the front of the allocation unit. Large files and inactive files are migrated towards the back of the allocation unit. A file is considered inactive if the access time on the inode is more than 14 days old. The time interval can be varied using the `-a` option. This reorganization should reduce the average seek time by placing inodes and frequently used data closer together.

The `fsadm` utility will try performing these reorganizations on all inodes on the file system. Each pass through the inodes will move the file system closer to the organization considered optimal by the `fsadm` utility. The first pass might place a file into one contiguous extent. The second pass might move the file into the same allocation unit at its inode. Then, since the first file has been moved, a third pass might move a file from another allocation unit into the space vacated by the first file during the second pass.

NOTE

`fsadm` is invoked as `/etc/fs/vxfs/fsadm` or `/sbin/fsadm`. This should change to the same invocation mechanism as other file system specific utilities in the future.

FILES

`lost+found/.fsadm`
: Lock file

SEE ALSO

`mkfs`(1M-vxfs), `fcntl`(2), `fs`(4-vxfs), `vxfsio`(7).

System Administrator's Reference Manual

fscat(1M)
VxFS

NAME

`fscat -` cat a Veritas File System (VxFS)

SYNOPSIS

`fscat [ –F vxfs ] [ –o` *offset* `] [ –l`*length* `] [ –b` *block_size* `]` *special*

DESCRIPTION

The `fscat` utility provides an interface to a Veritas snapshot file system similar to that provided by the `dd` utility invoked on the special file of other Veritas file systems. On most Veritas file systems, the block or character special file for the file system provides access to a raw image of the file system, for purposes such as backing up the file system to tape; on a snapshot file system access to the corresponding block or character special provides little useful information. The `fscat` utility, however, provides a stream of bytes representing the file system snapshot. This can be processed in a pipeline, written to a tape, etc. `fscat` will work when executed on the special device of any Veritas file system.

By default, the output is a stream of bytes that starts at the beginning of the file system and goes to the end. On a snapshot file system, data is read from the file system using the `VX_SNAPREAD` `ioctl(2)` on the mount point, in `VX_MAXIO` chunks. On other Veritas file systems, data is read from the specified special file in `VX_MAXIO` chunks. Unless otherwise specified, data is written in blocks of size `st_blksize` as returned by a `fstat` on stdout.

All numbers entered as options may have "0" prepended to indicate octal, or "0x" prepended to indicate hexadecimal. A "b" may be appended to indicate the value is in 512-byte blocks, a "k" to indicate the value is in kilobytes, or an "m" to indicate the value is in megabytes. Trailing letters must be joined to the numbers without any spaces.

OPTIONS

`–F vxfs`
: Specify a Veritas *FSType*.

`–o` *offset*
: Specify the starting offset, in bytes.

`–l` *length*
: Specify the length, in bytes. A *length* of 0 goes to the end of the file system.

`–b` *block_size*
: Specify the output block size, in bytes. *block_size* must be less than or equal to 1 megabyte.

NOTES

`fscat` is invoked as `/usr/lib/fs/vxfs/fscat`. This should change to the same invocation mechanism as other file system specific utilities in the future.

A snapshot file system cannot be written to. A snapshot file system exists only as long as it is mounted; once unmounted the special file no longer contains a snapshot file system.

`VX_MAXIO` is defined in `sys/fs/vx_param.h`.

SEE ALSO

`dd(1)` , `fs(4-vxfs)` , `vxfsio(7)` .

System Administrator's Reference Manual

fsck(1M)

NAME

`fsck(generic)` - check and repair file systems

SYNOPSIS

`fsck` [−F *FSType*] [−V] [−m] [*special ...*]

`fsck` [−F *FSType*] [*generic_options*] [−o *specific_options*] [*special ...*]

DESCRIPTION

`fsck` audits and interactively repairs inconsistent conditions for file systems. If the file system is inconsistent the user is prompted for concurrence before each correction is attempted. It should be noted that some corrective actions will result in some loss of data. The amount and severity of data loss may be determined from the diagnostic output. The default action for each correction is to wait for the user to respond "yes" or "no". If the user does not have write permission `fsck` defaults to a "no" action.

The file system should be unmounted when `fsck` is used. If this is not possible, care should be taken that the system is quiescent and that it is rebooted immediately afterwards if the file system is essential for the boot process.

special represents a block or character special device (e.g. `/dev/ios0/rsdisk000s8`). It is preferable that a character special device be used. `fsck` will not work on a block device if it is mounted. If *special* is not supplied, `fsck` looks through `/etc/vfstab` and executes `fsck` for all character specials in the "fsckdev" field of `/etc/vfstab` for which there is a numeric entry in the "fsckpass" field [see `vfstab(4)`].

If the −p option is given and no special devices are present, then `fsck` will preen the file system. In this case, `fsck` will use the information gathered from `/etc/vfstab` to inspect groups of disks in parallel to take maximum advantage of I/O overlap when checking the file systems. This causes `fsck` to run as quickly as possible. If the −s option is used in conjunction with the −p option, `fsck` will inspect the following specified number of filesystems per disk spindle. If the −s option is not used, `fsck` with the −p option defaults to inspecting up to 2 filesystems per disk spindle.

Note:

If disks are to be checked in parallel, the entry in the "fsckpass" field in `/etc/vfstab` for those disks to be checked in parallel is to be set to the same value ≥1.

If sent a `QUIT` signal, `fsck` will finish the file system checks, then exit with an abnormal return status. This is intended to cause an automatic reboot to fail, giving control back to the user.

OPTIONS

There are three types of option: *generic_options*, *other_options* and *specific_options*.

generic_options

Indicate options that are supported not only by the generic but also by the *FST*-specific module of the command (if that command exists).

other_options

Indicate options that are supported only by the generic command.

specific_options

Indicate suboptions that are supported only by the *FSType*-specific module [see `fsck(1M-ufs)` or `fsck(1M-vxfs)`] of the command. They are specified in a comma-separated list and/or as keyword-attribute pairs for interpretation by the *FSType*-specific module of the command. These suboptions are specified via the −o option.

The *generic_options* are:

−F *FSType*

Specifies the file system type on which to operate on the given *special* device. If no *special* devices are

given, then `fsck` will restrict its checks to file systems of that FSType.

The *FSType* should either be specified here or be determinable from `/etc/vfstab` by matching the *special* with an entry in the table. For legal *FSTypes* see `df(1)` .

`-V` Echoes the complete command line, but does not execute the command. The command line is generated by using the options and arguments provided by the user and adding to them information derived from `/etc/vfstab`. This option should be used to verify and validate the command line. When used with the *FSType* option, it passes the `-V` to the *FSType*-specific module for interpretation.

`-I` Insists that a file system be checked. This forces `fsck` to ignore the state of the file system and perform the consistency checks.

`-y` Assumes a yes response to all questions asked by `fsck`.

`-n` Assumes a no response to all questions asked by `fsck`.

`-p` Checks the file system in preen mode. Correct inconsistencies that can be fixed automatically, that is, inconsistencies that are deemed harmless and can be fixed without confirmation by the administrator. Examples of such inconsistencies are unreferenced i-nodes, incorrect counts in the superblocks, and missing blocks in the free list. If given without arguments, this option will sort all file systems that are to be checked for efficient throughput.

`-s` Specifies the number of filesystems to be inspected simultaneously per disk spindle. This option is only used with the `-p` option. By default, `fsck` with the `-p` option will inspect up to two filesystems per disk spindle at the same time. This can lead to degraded performance, however, if the filesystems are physically separated on the disk medium due to the cost of seeking. Valid values for the number of filesystems to inspect are 1 through 4.

`-l` *level*

Limits the action to file systems at a specified *level*. Legal levels are `0` through `9` and `u`:

Level `0`:
Essential file systems for boot process (such as `/usr`, `/var`)

Level `1-9`:
Other file systems.

Level `u`:
Check all levels except level 0.

`-o` Specifies *FSType*-specific options.

The *other_options* are:

`-m` Check but don't repair. This option checks that the file system is suitable for mounting.

NOTES

This command may not be supported for all *FSTypes*.

Certain utilities can cause a file system to be marked consistent when they are not. For example, `unlink`, `clri`, `mvdir`, and `fsdb` were designed to structurally modify the file system components at a low level, and so do not update any of the file system resource allocation maps or other related information. Consequently, use of these commands may leave a file system inconsistent, even when it is not marked as such. Any file system that is altered using one of these commands should always be checked using the `-I` option.

FILES

`/etc/vfstab`
list of default parameters for each file system

SEE ALSO

`df(1)` , `mkfs(1M)` , `vfstab(4)` .

FSType-specific modules: `fsck(1M-ufs)` , `fsck(1M-vxfs)` . Online call e.g.: "man 1M-ufs fsck".

System Administrator's Reference Manual

fsck(1M-ufs)

NAME

`fsck(ufs)` - file system consistency check and interactive repair

SYNOPSIS

`fsck` [`–F ufs`] [*generic_options*] [*special ...*]

`fsck` [`–F ufs`] [*generic_options*] [`–o b=#,w,r`] [*special ...*]

DESCRIPTION

`fsck` audits and interactively repairs inconsistent conditions on file systems. In this case, it asks for confirmation before attempting any corrections. Inconsistencies other than those mentioned below can often result in some loss of data. The amount and severity of data lost can be determined from the diagnostic output.

`fsck` corrects innocuous inconsistencies such as: unreferenced inodes, too-large link counts in inodes, missing blocks in the free list, blocks appearing in the free list and also in files, or incorrect counts in the super block, automatically. It displays a message for each inconsistency corrected that identifies the nature of, and file system on which, the correction is to take place. After successfully correcting a file system, `fsck` prints the number of files on that file system, the number of used and free blocks, and the percentage of fragmentation.

The default action for each correction is to wait for the operator to respond either `yes` or `no`. If the operator does not have write permission on the file system, `fsck` will default to a `–n` (no corrections) action.

Inconsistencies checked are as follows:

- Blocks claimed by more than one inode or the free list.
- Blocks claimed by an inode or the free list outside the range of the file system.
- Incorrect link counts.
- Incorrect directory sizes.
- Bad inode format.
- Blocks not accounted for anywhere.
- Directory checks, file pointing to unallocated inode, inode number out of range, absence of " . " and " . . " as the first two entries in each directory.
- Super Block checks: more blocks for inodes than there are in the file system.
- Bad free block list format.
- Total free block and/or free inode count incorrect.

Orphaned files and directories (allocated but unreferenced) are, with the operator's concurrence, reconnected by placing them in the `lost+found` directory. The name assigned is the inode number prepended with the following characters "`i-`". If the `lost+found` directory does not exist, it is created. If there is insufficient space its size is increased.

A file system may be specified by giving the name of the block or character special device on which it resides, or by giving the name of its mount point.

OPTIONS

`–F ufs`
: Specifies the `ufs`-FSType.

generic_options
: Options supported by the generic `fsck` command.

`–o`
: Specifies the `ufs` file system specific suboptions. If invalid options are specified, a warning message is printed and the invalid options are ignored. The following options are available:

`b=#` Use the block specified as the super block for the file system. Block 32 is always an alternate super block.

`r` Read-only check of the file system. `fsck` does not attempt to fix and inconsistencies.

`w` Only checks those file systems that are designated as `read-write` in `/etc/vfstab`.

SEE ALSO

`crash`(1M) , generic `fsck`(1M) , `mkfs`(1M) , `vfstab`(4) , `fs`(4-ufs) .

System Administrator's Reference Manual

fsck(1M-vxfs)
VxFS

NAME

`fsck(vxfs)` - check and repair a Veritas File System (VxFS)

SYNOPSIS

`fsck` [`–F vxfs`] [*generic_options*] [`–o full,nolog`] *special* ...

DESCRIPTION

This manual page is an addendum to the `fsck(1M)` manual page. It describes the Veritas File System (VxFS) specific features of the generic `fsck(1M)` command.

OPTIONS

`–F vxfs`
Specifies the Veritas file system type (*FSType*).

generic_options
Options supported by the generic `fsck(1M)` command.

`–o` Specifies Veritas-specific options. These options can be a combination of the following in a comma-separated list:

`full` Performs a full file system check. The default is to perform an activity log replay only. Since the Veritas file system maintains an activity log, a complete check is generally not required. If the file system detects damage or the log replay operation detects damage, an indication that a complete check is required is placed in the super block, and a full check is performed.

`nolog`
Do not perform log replay. This option may be used if the log area was physically damaged.

When a full check is performed, the following inconsistencies are checked:

- Blocks claimed by more than one inode or the free list.
- Blocks claimed by an inode outside the range of the file system.
- Incorrect link counts.
- Size checks:
  - Incorrect number of blocks.
  - Directory entry format.
- Bad inode format.
- Blocks not accounted for anywhere.
- Directory checks:
  - File pointing to unallocated inode.
  - Inode number out of range.
  - Linkage to parent directory.
  - Hash chain linkage.
  - Free space count.
- Super block checks:
  - Checksum mismatch.
  - More blocks for inodes than there are in the file system.
- Bad free block list format.

- Total free block and/or free inode count incorrect.

Orphaned files and directories (allocated but unreferenced) are, with the user's concurrence, reconnected by placing them in the `lost+found` directory. The name assigned is the inode number. The only restriction is that the directory `lost+found` must already exist in the root of the file system being checked.

OUTPUT

Structural errors discovered during a full check are displayed on `stdout`. Responses required during a full check are read from `stdin`.

The following return codes are used for the `-m` (generic) option:

`0` The file system is unmounted and clean.

`32` The file system is unmounted and needs checking.

`33` The file system is mounted.

`34` The stat of the device failed.

Other
: The state could not be determined because of an error.

ERROR/DIAGNOSTICS

All error messages that relate to the contents of a file system produced during a log replay are displayed on `stdout`. All I/O failures and exit messages are displayed on `stderr`.

NOTES

A full file system check will always perform any pending extended inode operations without operator interaction. If a structural flaw is detected, the full `fsck` flag will be set on the file system, without operator interaction.

SEE ALSO

`crash`(1M), `fsck`(1M), `mkfs`(1M), `ncheck`(1M), `uadmin`(2), `fs`(4-vxfs).

System Administrator's Reference Manual

fsdb(1M)

NAME

`fsdb(generic)` - file system debugger

SYNOPSIS

`fsdb` [`-F` *FSType*] [`-V`] [`-z` *i-number*] *special*

DESCRIPTION

`fsdb` is a file system debugger which allows for the manual repair of a file system after a crash. *special* is a special device used to indicate the file system to be debugged. `fsdb` is intended for experienced users only. *FSType* is the file system type to be debugged. Since different *FSTypes* have different structures and hence different debugging capabilities the manual pages for the *FSType*-specific `fsdb` should be consulted for a more detailed description of the debugging capabilities.

The specified options are supported not only by the generic but also by the *FST*-specific module of the command.

OPTIONS

`-F` Specifies the *FSType* on which to operate. The *FSType* should either be specified here or be determinable from `/etc/vfstab` by matching the *special* with an entry in the table.

`-V` Echoes the complete command line, but does not execute the command. The command line is generated by using the options and arguments provided by the user and adding to them information derived from `/etc/vfstab`. This option should be used to verify and validate the command line. When used with the *FSType* option, it passes the `-V` to the *FSType*-specific module for interpretation.

`-z` *i-number*
Clears the i-node identified by *i-number*. Non-interactive.

`-o` Specifies *FSType*-specific options.

FILES

`/etc/vfstab`
List of default parameters for each file system

SEE ALSO

`mkfs`(1M) , `vfstab`(4) .

Manual pages for the *FSType*-specific modules of `fsdb`.

System Administrator's Reference Manual

fsdb(1M-ufs)

NAME

`fsdb(ufs)` - ufs file system debugger

SYNOPSIS

`fsdb` [`–F ufs`] [*generic_options*] *special* [–]

DESCRIPTION

`fsdb` can be used to patch up a damaged `ufs` file system after a crash. It has conversions to translate block and i-numbers into their corresponding disk addresses. Also included are mnemonic offsets to access different parts of an i-node. These greatly simplify the process of correcting control block entries or descending the file system tree.

`fsdb` contains several error-checking routines to verify i-node and block addresses. These can be disabled if necessary by invoking `fsdb` with the optional "`–`" argument or by the use of the `O` symbol. (`fsdb` reads the `i`-size and `f`-size entries from the superblock of the file system as the basis for these checks.)

OPTIONS

`–F ufs`
: Specifies the `ufs`-FSType.

generic_options
: Options supported by the generic `fsdb` command.

Numbers are considered decimal by default. Octal numbers must be prefixed with a zero. During any assignment operation, numbers are checked for a possible truncation error due to a size mismatch between source and destination.

`fsdb` reads a block at a time and will therefore work with raw as well as block I/O. A buffer management routine is used to retain commonly used blocks of data in order to reduce the number of read system calls. All assignment operations result in an immediate write-through of the corresponding block.

The symbols recognized by `fsdb` are:

`#` absolute address

`#i` convert from i-number to i-node address

`#b` convert to block address

`#f` convert to fragment address

`#f`*n* same as above, except the fragment is regarded as having length *n*, where *n* is either 1, 2, ..., `MAXFRAG` frags

`c#` convert from cylinder group summary number to cylinder group summary address

`d#` directory slot offset

`d#`*i* same as above, except instructs `fsdb` to skip to the inode *i* associated with the directory

`d#`*c* same as above, except instructs `fsdb` to skip to the cylinder group *c* associated with the directory

`+, –, *, /`
: address arithmetic

`q` quit

`>, <` save, restore an address

`=` numerical assignment

`=+` incremental assignment

`=–` decremental assignment

`="` character string assignment

`O` error checking flip flop

`p` general print facilities

`f` file print facility

`B` byte mode

`S` word (short) mode

`L` double word (int) mode

`I` i-node mode

`R` diRectory mode

`W` same as `S`

`D` same as `L`

`!` escape to shell

`?` print a help menu

`N` open a new file system for debugging

`C` print current specifications for the file system being debugged

`P` print current state of `fsdb` global parameters

`F` print state of current disk block buffers

The print facilities generate a formatted output in various styles. The current address is normalized to an appropriate boundary before printing begins. It advances with the printing and is left at the address of the last item printed. The output can be terminated at any time by typing the delete character. If a number follows the `p` symbol, that many entries are printed. A check is made to detect block boundary overflows since logically sequential blocks are generally not physically sequential. If a count of zero is used, all entries to the end of the current block are printed.

The print options available are:

`i` print as i-nodes

`d` print as directories

`s` print as cylinder group summary structures

`o` print as octal words

`x` print as hexadecimal words

`e` print as decimal words

`c` print as characters

`b` print as octal bytes

The `f` symbol is used to print data blocks associated with the current i-node. If followed by a number, that block of the file is printed. (Blocks are numbered from zero.) The desired print option letter follows the block number, if present, or the `f` symbol. This print facility works for small as well as large files. It checks for special devices and that the block pointers used to find the data are not zero.

Dots, tabs, and spaces may be used as function delimiters but are not necessary. A line with just a new-line character will increment the current address by the size of the data type last printed. That is, the address is set to the next byte, word, double word, directory entry or i-node, allowing the user to step through a region of a file system. Information is printed in a format appropriate to the data type. Bytes, words and double words are displayed with the octal address followed by the value in octal and decimal. A `.B`, `.W` or `.D` is appended to the address for byte, word (short) and double word (int) values, respectively. Directories are printed as a directory slot offset followed by the decimal i-number and the character representation of the entry name. I-nodes are printed with labeled fields describing each element.

The following mnemonics are used for i-node examination and refer to the current working i-node:

`md` mode

`ln` link count

`uid` user ID number
`gid` group ID number
`sz` file size
`bc` block count
`a#` data block numbers (0-13)
`db#` direct data block numbers (0-10)
`ib#` indirect data block numbers (11-13)
`at` access time
`mt` modification time
`ct` inode creation time
`maj` major device number
`min` minor device number

The basic syntax of `fsdb` is as follows:

[*item*][*offset*][*print*/*replace*]

where *item* is either a block, directory, cylinder group, or i-node number, *offset* is a mnemonic for an i-node offset or any arbitrary number of bytes, and *print*/*replace* will either print the specified item in the indicated style or replace the specified item with the specified value. Also note that a series of commands separated by carriage returns are functionally equivalent to the same series of commands when typed on a single line.

EXAMPLES

`386i` prints i-number 386 in an i-node format. This now becomes the current working i-node.

`ln=4` changes the link count for the working i-node to 4.

`ln=+1`
increments the link count by 1.

`fc` prints, in ASCII, block zero of the file associated with the working i-node.

`2i.fd`
prints the first 32 directory entries for the root i-node of this file system.

`d5i.fc`
changes the current i-node to that associated with the 5th directory entry (numbered from zero) found from the above command. The first logical block of the file is then printed in ASCII.

`8192B.p0o`
prints the superblock of this file system in octal.

`2i.a0b.d7=3`
changes the i-number for the seventh directory slot in the root directory to 3. This example also shows how several operations can be combined on one command line.

`d7.nm="`*name*"
changes the name field in the directory slot to the given string. Quotes are optional when used with `nm` if the first character is alphabetic.

`a2b.p0d`
prints the third block of the current i-node as directory entries.

NOTES

Careless use of `fsdb` will result in an inconsistent file system and may cause information to be lost. Furthermore, it may not be fully recoverable by `fsck`.

SEE ALSO

`fsck`(1M) , generic `fsdb`(1M) , `dir`(4-ufs) , `fs`(4-ufs) .

System Administrator's Reference Manual

fsdb(1M-vxfs)
VxFS

NAME

`fsdb(vxfs)` - Veritas File System debugger (VxFS)

SYNOPSIS

`fsdb` [`–F vxfs`] [*generic_options*] *special*

DESCRIPTION

This manual page is an addendum to the `fsdb(1M)` manual page. It describes the Veritas File System (VxFS) specific features of the `fsdb` command.

`fsdb` can be used to patch up a damaged Veritas file system. It has conversions to translate blocks and inumbers into their corresponding disk addresses. Also included are mnemonic offsets to access different parts of an inode. These greatly simplify the process of correcting control block entries or descending the file system tree.

OPTIONS

`–F vxfs`
: Specifies the Veritas file system type (*FSType*).

generic_options
: Options supported by the generic `fsdb` command.

By default, numbers are considered decimal. Octal numbers must be prefixed with a zero. Hexadecimal numbers must be prefixed with `0x`. When using hexadecimal numbers, it is preferable to follow the number with a space, since a number of commands are letters that are also hexadecimal digits. In this document, a pound sign (#) is used to indicate that a number is to be specified.

`fsdb` reads a block at a time and will work with raw and block I/O. A buffer management routine is used to retain commonly used blocks of data in order to reduce the number of read system calls. All assignment operations result in an immediate write-through of the corresponding block.

The following symbols are recognized by `fsdb`:

`? h help`
: Print command summary.

`!` Escape to shell.

`q` Quit.

`"string"`
: A character string. Inside of a character string, a NULL may be specified with `\0` and a double quote may be specified with a `\"` and a backslash may be specified by `\\`.

`- + * /`
: Add, subtract, multiply, and divide.

`=` Assignment.

`i` An inode.

`au` An allocation unit.

`b` Ablock.

`im` The immediate data area of an inode. Small files (96 bytes or less) may be stored directly in the inode itself, in the area normally occupied by data block numbers and extent sizes.

`cdb` Current directory block.

`d` A directory entry.

`a` An inode address entry.

`B` Abyte.

`H` A half-word (2 bytes).

`W` A word (4 bytes).

`D` A double-word (8 bytes).

`p` General print facility.

`calc` Simple calculator and base converter.

`find` Find a matching pattern in the file system.

The print facility recognizes the following print formats:

`S` Print as a super block.

`A` Print as an allocation unit header.

`L` Print as intent log records.

`I` Print as inodes.

`dent` Print as directory entries.

`db` Print as a directory block.

`dh` Print as a directory header.

`o` Print as octal words.

`oB oH oW oD`
Print as octal bytes, half-words, words, or double-words.

`x` Print as hexadecimal words.

`xB xH xW xD`
Print as hexadecimal bytes, half-words, words, or double-words.

`e` Print as decimal words.

`eB eH eW eD`
Print as decimal bytes, half-words, words, or double-words.

`c` Print as characters.

Changes to inode fields may be made symbolically. The following symbols represent inode fields:

`md` Inode mode field.

`ln` Inode link count field.

`uid` Inode user ID number field.

`gid` Inode group ID number field.

`sz` Inode file size field.

`de #` Inode direct extent data block numbers (0 – 9).

`e #` Inode direct extent sizes (0 – 9).

`ie #` Inode indirect extent data block numbers (0 – 1).

`ies #`
Inode indirect extent size.

`im` Immediate data area. Small files (96 bytes or less) may be stored directly in the inode itself, in the area normally occupied by data block numbers and extent sizes.

`at` Inode access time field.

`ct` Inode change time field.

`mt` Inode modification time field.

`af` Inode allocation flags field.

`gen` Inode generation count field.

`org` Inode mapping type field (`di_orgtype`).

`fe` Inode fixed extent size field.

`bl` Inode blocks held field.

`eopflg`
Inode extended operation flag field.

`eopdat`
Inode extended operation data field.

`maj` If device, inode major number.

`min` If device, inode minor number.

`pd` If directory, inode parent directory.

`res` If regular file, inode reservation.

`serhi`
Inode high order word of serial number (`di_hserial`).

`serlo`
Inode low order word of serial number (`di_lserial`).

Changes to directory block fields may be made symbolically. The following symbols represent directory block fields:

`tfree`
Total free space (only if in a data block).

`hash #`
Hash chain start (0 through 31, only if in a data block).

`nhash`
Number of hash chains.

`d #` Directory entry (variable number of entries).

Changes to directory entry fields may be made symbolically. The following symbols represent directory entry fields:

`ino` Inode number.

`nm` Entry name.

`nmlen`
Name length.

`nmmax`
Maximum name length (only if in immediate area).

`reclen`
Record length (only if in a data block).

`hnext`
Name hash next (only if in a data block).

It is preferable to separate each token on a command line with a space. Although the command parser does not insist on space separation, there is no ambiguity in the command language if each token is separated with a space. For example, the command "`0x23b b`" sets the current position to block 0x23b hexadecimal. The command "`0x23bb`" is invalid, since the command is parsed as simply a hexadecimal number. The command "`23b`" positions to block 23 decimal, since the command is not ambiguous.

Commands are separated by new lines or multiple commands may be placed on one line, separated by a period (.) or a semicolon (;). When multiple commands are placed on one line, generally only the last command displays results. This allows positioning commands to be followed by printing commands or change commands without intermediate printing. The following commands are supported:

`# B`|H|W|D
Set current position in the file system to the specified offset in bytes, half-words, words, or double-words. If the last command on a line, print the byte, half-word, word, or double-words in hexadecimal.

+|- # B|H|W|D
Set current position to specified relative offset in bytes, half-words, words, or double-words. If the last command on a line, print the byte, half-word, word, or double-words in hexadecimal.

`# b` Set current position in the file system to the specified offset in blocks. Set current block position to the resulting offset. The block size is the block size of the file system. If the last command on a line, print the first word in the block in hexadecimal.

+|- # b
Set current position to specified relative offset in blocks. Set current block position to the resulting offset. If the last command on a line, print the first word in the block in hexadecimal.

`b` Set current position to current block position (the block specified by the last [+|-] `# b` operation). If the last command on a line, print the first word in the block in hexadecimal.

`# i` Set current position in the file system to the `ilist` entry for the specified inode. Set current inode position to the resulting offset. If the last command on a line, print the `ilist` entry for the inode.

+|- # i
Set current position in the file system to the `ilist` entry for the specified relative inode. Set current inode position to the resulting offset. If the last command on a line, print the `ilist` entry for the inode.

`i` Set current position in the file system to the current inode position. If the last command on a line, print the `ilist` entry for the inode.

`a #` Set current position to specified offset in blocks specified by the inode address #. Addresses 0 through 9 are for direct extents (de). Addresses 10-11 are for indirect extents (ie). The addresses are displayed when printing an `ilist` entry. Set current block position to the resulting offset. If the last command on a line, print the first word in the block in hexadecimal.

`im` Set current position to immediate data area of the current inode. Set current block position to the resulting offset. If the last command on a line, print the first word in the block in hexadecimal.

`# B`|H|W|D = # [#]
Set the current position and change the number at the specified offset to the given number. If a double-word offset is specified, then two numbers separated by a space are required. The resulting value is printed in hexadecimal.

+|- # B|H|W|D = # [#]
Set the current position and change the number at the specified relative offset to the given number. If a double-word offset is specified, then two numbers separated by a space are required. The resulting value is printed in hexadecimal.

`# B`|H|W|D = "*string*"
Set the current position and change the characters at the specified offset to the given string. The resulting value is printed as a character string.

+|- # B|H|W|`D` = "*string*"
Set the current position and change the characters at the specified relative offset to the given string. The resulting value is printed as a character string.

`p` [#] format
Print the contents of the file system at the current offset as the specified number of entries of a given format. The allowable print formats are specified above. If a number of entries to print is not specified, one entry is printed.

`inode_field = #`
Set the contents of the given inode field to the specified number. The current inode specifies the inode list entry to be modified. The symbols representing inode fields are previously listed.

`directory_block_field = #`
Set the contents of the given directory block field to the specified number. The current block is treated as a directory block and the offset in that block which is represented by the given field is changed. The symbols representing directory block fields are listed above.

`d #` Set the current directory entry to the specified number. The current block is treated as a directory block.

If the current block is an immediate data area for an inode, then the block is treated as containing immediate directory entries. If the last command on a line, the directory entry at the resulting offset is printed.

`directory_entry_field = #`
Set the contents of the given directory field to the specified number. The current directory entry specifies where the directory entry is located. The resulting value is printed in hexadecimal.

`nm` = "*string*"
Set the directory name field of the current directory entry to the specified string. The resulting value is printed as a character string.

`calc # [+|-|*|/ #]`
Take a number or the sum, difference, product or dividend of two numbers and print in decimal, octal, hexadecimal and character format.

`find # B`|H|W|D [#]
Search for the given numeric pattern in the file system. The size of the object to match is specified. If a double-word is specified, then two numbers must be given. The search is performed forward from the current offset. A maximum number of blocks to search may be specified. If found, the location and value are printed in hexadecimal.

`find # B`|H|W|D "*string*"
Search for the given character string in the file system. The search is performed forward from the current offset. A maximum number of blocks to search may be specified. If found, the location and string are printed.

`fmtlog`
Format all intent log entries. A completely formatted intent log can be quite lengthy. It is a good idea to use `fsdb` as a filter and redirect the output to a file or pager to look at a complete log format.

EXAMPLES

`386i` Prints inumber 386 in an inode format. This now becomes the current working inode.

`ln=4` Changes the link count for the working inode to 4.

`1024B.pS`
Prints the super block of this file system symbolically.

`1024B.p0o`
Prints the super block of this file system in octal.

`2i.a0b.d7.ino=3`
Changes the inumber for the seventh directory slot in the `root` directory to 3. This example also shows how several operations can be combined on one command line.

`d7.nm="`*name*"
Changes the name field in the directory slot to the given string.

`23i.im.pdb`
Prints the immediate area of inode 23 as a directory block.

`23i.im.d5`
Prints sixth directory entry in the immediate area of inode 23.

SEE ALSO

`fsck`(1M), `fsdb`(1M), `fs`(4-vxfs).

System Administrator's Reference Manual

fsminfo(1M)

NAME

`fsminfo` - report information about mounted file systems

SYNOPSIS

`fsminfo [ –p ]` *device ...*

`/sbin/fsminfo [ –p ][ –ln ]`

DESCRIPTION

`fsminfo` prints information about mounted file systems. This command differs from the `mount(1M)` command in that it queries the kernel rather than relying on `/etc/mnttab`. Therefore, it is more reliable. It can also be used to reconstruct `/etc/mnttab` if the latter becomes lost or corrupted.

Other than stated below, the exit status is 0 unless there was an error, in which case it will be 255.

OPTIONS

No option specified:

If no options are given, then `fsminfo` expects one or more arguments and will check if the given device is mounted or if the given directory is mounted upon. A message is printed stating the above and the exit status is the number of arguments mounted or mounted upon.

`–p` Prints the output in a format that is easily parsed (the default is a more human-readable format).

`–n` Prints the number of mounted file systems.

`–l` Lists all mounted file systems. The output is in the following form:

```
Device            Size    Type         FSType MountFlags  MountPoint
```

Each column should be self-explanatory.

`Size` is in 512-byte blocks and is applicable only to disk based file systems.

`MountFlags` specifies only the following (generic) mount flags: `ro`, `rw` and `nosuid`.

EXAMPLE

```
fsminfo –l
```

```
# Device                   Size   Type       FSType MountFlags  MountPoint
/dev/ios0/sdisk000s0      25800  local         ufs <rw>         /
                              0 applic      namefs <rw>         [unknown]
/dev/ios0/sdisk022s3     568890  local         vxfs <rw>         /home/nerv
                        1017226 remote          nfs <ro>         /pen/svr4
                         512316 remote          nfs <ro>         /misc/mesh
                              0 applic      namefs <rw>         [unknown]
/dev/vd/vdisk1             7168  local         ufs <rw>         /vd1
/dev/ios0/sdisk074s8    2059776  local         vxfs <rw>         /puls/3
/dev/ios0/sdisk000s4      97500  local         ufs <rw>         /var
/dev/ios0/sdisk000s3     204900  local         ufs <rw>         /usr
/dev/ios0/sdisk000s2     168000  local         ufs <rw>         /opt
/dev/fd                       0 applic        fdfs <rw>         /dev/fd
/proc                         0 applic        proc <rw>         /proc
```

SEE ALSO

`df(1)`, `mount(1M)`, `fstinfo(1M)`, `sysfs(2)`, `vfstab(4)`, `mnttab(4)`.

System Administrator's Reference Manual

fstinfo(1M)

NAME

`fstinfo -` report information about installed file system types

SYNOPSIS

`fstinfo [ -p ] –t [` *attributes* `]`

`fstinfo [ -p ] –F` *FSType*

DESCRIPTION

The system is configured with a number of file system types (FSTypes). Some of these are used for data storage (local or remote), others are used to access kernel data structures and a few are used internally by the system. In the future, other file system types may become available that provide a different service.

Due to this proliferation of different FSTypes, it becomes convenient to group FSTypes by certain attributes. One can then obtain specific information given an attribute or an FSType. `fstinfo` provides an interface to do just that. It prints the attributes of the installed file system types (FSTypes).

Currently, *attributes* are divided into two categories: "allocation" attributes and "accessibility" attributes.

The allocation *attributes* will be either `disk-based` or `kernel-based`. A `disk-based` file system type means there are physical blocks in the file system. A `kernel-based` file system type has no (user-visible) physical storage. (Note that a memory disk qualifies as physical storage as far as file systems are concerned.)

The accessibility *attributes* are dependent upon the allocation *attributes* as follows:

`disk-based`
: The *attributes* will be either `local` or `remote`.

`kernel-based`
: The *attributes* will be either `internal` or `application`.

`disk-based` file system types hold user data on a physical disk somewhere. The disk space can either be `local` or `remote`. Examples of these *FSTypes* are: `ufs` and `nfs`.

`kernel-based` file system types are not oriented around manipulating physical storage. Rather, they provide a file-system-like interface to other objects. They are currently categorized by whether they have a user interface or not:

- `kernel-based application` file system types are those that have a user-level interface and/or utilities. These file systems need to be mounted via the `mount(1M)` utility. Examples of these *FSTypes* are: `proc` and `fdfs`.
- `kernel-based internal` file system types are those that are of interest only to the kernel and do not have user level interfaces or utilities. They are automatically "mounted". Examples of these *FSTypes* are: `specfs` and `fifofs`.

OPTIONS

`–p` Prints the output in a format that is easily parsed (the default is a more human-readable format).

`–t` Lists all installed FSTypes that have the given *attributes*. If no *attributes* are given, show all FSTypes and their *attributes*. The exit status is the number of FSTypes that match the given *attributes*.

`–F` Displays the attributes of the given *FSType* (FSType names are obtainable via the `–t` option).

EXAMPLES

```
fstinfo -t
fdfs:           kernel-based application filesystem.
fifofs:         kernel-based internal    filesystem.
namefs:         kernel-based application filesystem.
nfs:            disk-based   remote      filesystem.
```

```
proc:            kernel-based application filesystem.
sockfs:          kernel-based internal    filesystem.
specfs:          kernel-based internal    filesystem.
ttyfs:           kernel-based internal    filesystem.
ufs:             disk-based   local       filesystem.
vxfs:            disk-based   local       filesystem.
fstinfo -t disk-based local remote
nfs:             disk-based   remote      filesystem.
ufs:             disk-based   local       filesystem.
vxfs:            disk-based   local       filesystem.
fstinfo -t kernel-based
fdfs:            kernel-based application filesystem.
fifofs:          kernel-based internal    filesystem.
namefs:          kernel-based application filesystem.
proc:            kernel-based application filesystem.
sockfs:          kernel-based internal    filesystem.
specfs:          kernel-based internal    filesystem.
ttyfs:           kernel-based internal    filesystem.
fstinfo -F vxfs
vxfs:            disk-based   local       filesystem.
fstinfo -F proc
proc:            kernel-based application filesystem.
```

DIAGNOSTICS

Other than stated above, the exit status is 0 unless there was an error, in which case it will be 255.

NOTES

The internal kernel-based file system types should really be kept hidden from users (they are completely internal to the system and are of no interest to users). However, since other utilities and system calls refer to them, they are included here.

SEE ALSO

df(1), fsminfo(1M), mount(1M), sysfs(2), vfstab(4).

System Administrator's Reference Manual

fstyp(1M)

NAME

`fstyp -` determine file system type

SYNOPSIS

`fstyp [ –v ]` *special*

DESCRIPTION

`fstyp` allows the user to determine the file system type of unmounted file systems using heuristic programs.

An `fstyp` module for each file system type to be checked is executed; each of these modules applies some appropriate heuristic to determine whether the supplied *special* file is of the type for which it checks. If it is, the program prints on standard output the usual file-system identifier for that type and exits with a return code of 0; if none of the modules succeed, the error message `unknown_fstyp (no matches)` is returned and the exit status is 1. If more than one module succeeds the error message `unknown_fstyp (multiple matches)` is returned and the exit status is 2.

OPTIONS

`–v` Produces verbose output. This is usually information about the file systems superblock and varies across different *FSTypes*.

LOCALE

The `LC_MESSAGES` environment variable governs the language in which message texts are displayed.

If `LC_MESSAGES` is undefined or is defined as the null string, it defaults to the value of `LANG`. If `LANG` is likewise undefined or null, the system acts as if it were not internationalized.

If any of the locale variables has an invalid value, the system acts as if none of the variables was set.

The `LC_ALL` environment variable governs the entire locale. `LC_ALL` takes precedence over all the other environment variables that affect internationalization.

NOTES

The use of heuristics implies that the result of `fstyp` is not guaranteed to be accurate.

System Administrator's Reference Manual

fstyp(1M-vxfs)

NAME

`fstyp(vxfs)` - check for Veritas superblock as identifying sign offile system

SYNOPSIS

`fstyp` [–v] *device*

DESCRIPTION

This is an addendum to the `fstyp(1M)` manual page. It describes the Veritas File System (VxFS) specific features of the `fstyp` command.

The file system-specific versions of the utility check for identifying signs of a file system. If the appropriate file systems are present, the utility will display the type string for the file system on standard output, and return an exit code of zero. If the appropriate file system is not found, a non-zero exit code is returned.

The `vxfs` file system-specific version of the `fstyp` utility checks for an `vxfs` superblock. Specifically, the `fs_magic` field of the superblock is checked to see if it is `VX_MAGIC`. The format of the superblock is specified in `fs`(4) (VxFS). If an `vxfs` file system is found, `vxfs` is displayed on standard output.

OPTIONS

`-v` The superblock fields, the number of free blocks and inodes, and the number of free extents by size are also displayed.

SEE ALSO

`fstyp(1M)` , `fs(4-vxfs)` .

System Administrator's Reference Manual

fuser(1M)

NAME

`fuser` - identify processes using a file or file structure

SYNOPSIS

`/usr/sbin/fuser` [–[c|f]ku] [–*signal*] *files* | *resources* {[[–] | [–[c|f]ku] [–*signal*]} *files* | *resources*] ...

DESCRIPTION

`fuser` outputs the process IDs of the processes that are using the *files* or remote *resources* specified as arguments. Each process ID is followed by a letter code, interpreted as follows: if the process is using the file as

1) its current directory, the code is `c`,
2) its root directory, the code is `r`,
3) an open file, the code is `o`, or
4) its text file, the code is `t`.

For block special devices with mounted file systems, all processes using any file on that device are listed. For remote resource names, all processes using any file associated with that remote resource (Remote File Sharing) are reported. For all other types of files (text files, executables, directories, devices, etc.) only the processes using that file are reported.

OPTIONS

`–c` May be used with files that are mount points for file systems. With that option the report is for use of the mount point and any files within that mounted file system.

`–f` When this is used, the report is only for the named file, not for files within a mounted file system.

`–k` Unless changed with –*signal* the `SIGKILL` signal is sent to each process. Since this option spawns kills for each process, the kill messages may not show up immediately [see `kill(2)`].

`–u` The user login name, in parentheses, also follows the process ID.

–*signal*
Use the specified signal instead of `SIGKILL` when killing processes. Signals can be specified either by name [e.g. `–HUP`] or by number [e.g. `–1`; cf. `signal(5)`].

If more than one group of files are specified, the options may be respecified for each additional group of files. A lone dash cancels the options currently in force.

The process IDs are printed as a single line on the standard output, separated by spaces and terminated with a single new line. All other output is written on standard error.

Any user with permission to read `/dev/kmem` and `/dev/mem` can use `fuser`. Only the superuser can terminate another user's process.

NOTES

If an RFS resource from a pre Release 4 server is mounted, `fuser` can only report on use of the whole file system, not on individual files within it.

Because `fuser` works with a snapshot of the system image, it may miss processes that begin using a file while `fuser` is running. Also, processes reported as using a file may have stopped using it while `fuser` was running. These factors should discourage the use of the `–k` option.

FILES

`/stand/unix`
for system namelist

`/dev/kmem`
for system image

`/dev/mem`
also for system image

SEE ALSO

`ps`(1), `mount`(1M), `kill`(2), `signal`(2), `signal`(5).

System Administrator's Reference Manual

fw_check(1M)
RM600 only

NAME

`fw_check -` consistency checking of MBII and SP bus controllerfirmware

SYNOPSIS

`fw_check [-l][-h]`

DESCRIPTION

`fw_check` checks the firmware versions of the individual MBII and SP bus controllers and the EIPs. If there is an inequality in controllers of the same type, this is reported and if necessary, the controller is flagged as defective.

The current values of the hardware and firmware versions are returned by `showconf(8)` . These values serve as a basis for the checks. The controllers are also checked for minimum values (apart from EIPs).

The check extends to both the firmware versions and the hardware versions.

The comparison values are contained in a reference file `fw_ref` and derived by the Offline Flash Utility using the `mcs -p flsh_b20` command. This `fw_ref` file is in the `/etc/firm/fwref.d` directory.

In addition, all files in this directory with the `fw_ref` format and whose names begin with `fw` are included in the check. Additional reference values can be specified in this way.

OPTIONS

`-l` No entry is made in the log if this option is specified. The `/var/lar/hwconfig/HW_CONFIG` file is not updated.

An entry is made in the log if this option is not specified. If necessary, the "error" attribute is set in the `/var/lar/hwconfig/HW_CONFIG` file with the reason: Firmware inconsistency.

`-h` A help text relating to `fw_check` is output.

INTEGRATION INTO THE SYSTEM

1. Direct call.
2. When starting up the system.

   The `fw_check` call is integrated into the `rc2` initphase. Entries are then made in the log and `/var/lar/hwconfig/HW_CONFIG` is updated.
3. Call from the Service menu.

   `fw_check` is integrated into the Service menu.
   - Operating system
   - Configuration
   - Firmware Consistency Check

   No entries are made in the log and `/var/lar/hwconfig/HW_CONFIG` is not updated.

DIAGNOSTICS

The error messages have the following format:

`fw_check:`*`errno`*`:cab_id:`*`cabno`*`:slot:`*`slot`*`:chan:`*`channo`*`:bus:`*`bustype`*`:`□*`errortext`*

For EIP messages:

`fw_check:`*`errno`*`:cab_id:`*`cabno`*`:hw_rev:`*`hardware-rev.`*`:cab_type:`*`cabinettype`*`:\`
`serial:`*`EIP_serialno`*`:`□*`errortext`*

The error text is self-explanatory. The text contains as required:

`module`
: Product name of the controller

`firmware rev.`
: Firmware revision

`hardware rev.`
: Hardware revision

The controller that triggered the error message is specified at the start of the error message.

If there is only one controller with a later firmware version and many controllers with a lower version, that are however the same, all these lower versions are declared defective. The problem can either be solved by bringing all the controllers up to a higher version or the one controller down to the lower version, if this version is valid.

You can get a complete overview of the configuration with the `showconf` command or with `showconf -c` for EIPs.

More information relating to the error messages and the corresponding responses is contained in the error descriptions that can be requested using the `getemm fw_check:`*errno* command.

FILES

`/var/lar/hwconfig/HW_CONFIG`
: Contains data relating to the hardware configuration

`/etc/firm/fwref.d/fw_*`
: Reference files with firmware estimated values

SEE ALSO

`mcs`(1), `getemm`(1M), `rc2`(1M), `showconf`(8).

System Administrator's Reference Manual

fwtmp(1M)

NAME

`fwtmp, wtmpfix -` manipulate connect accounting records

SYNOPSIS

`/usr/lib/acct/fwtmp [ –ic ]`

`/usr/lib/acct/wtmpfix [` *files* `]`

DESCRIPTION

`fwtmp` reads from standard input and writes to standard output, converting binary records of the type found in `/var/adm/wtmp` to formatted ASCII records. The ASCII version is useful when it is necessary to edit bad records.

The argument `–ic` is used to denote that input is in ASCII form, and output is to be written in binary form.

`wtmpfix` examines standard input or named files in `utmp.h` format, corrects the time/date stamps to make the entries consistent, and writes to standard output. A – can be used in place of *files* to indicate standard input. If time/date corrections are not performed, `acctcon` will fault when it encounters certain date-change records.

Each time the date is set, a pair of date change records are written to `/var/adm/wtmp`. The first record is the old date denoted by the string "`old time`" placed in the `line` field and the flag `OLD_TIME` placed in the `type` field of the `utmp` structure. The second record specifies the new date and is denoted by the string `new time` placed in the `line` field and the flag `NEW_TIME` placed in the `type` field. `wtmpfix` uses these records to synchronize all time stamps in the file.

In addition to correcting time/date stamps, `wtmpfix` will check the validity of the `name` field to ensure that it consists solely of alphanumeric characters or spaces. If it encounters a name that is considered invalid, it will change the login name to `INVALID` and write a diagnostic to standard error. In this way, `wtmpfix` reduces the chance that `acctcon` will fail when processing connect accounting records.

FILES

`/var/adm/wtmp`

`/usr/include/utmp.h`

SEE ALSO

`acctcom(1)`, `ed(1)`, `acct(1M)`, `acctcms(1M)`, `acctcon(1M)`, `acctmerg(1M)`, `acctprc(1M)`, `acctsh(1M)`, `runacct(1M)`, `acct(2)`, `acct(4)`, `utmp(4)`.

genal(1M) - grpck(1M)

System Administrator's Reference Manual

genal(1M)

NAME

`genal -` program for generating the German language exception list

SYNOPSIS

`/usr/bin/genal`

DESCRIPTION

The `genal` command generates a new exception list for German hyphenation. A user with the appropriate permissions can edit the exception list file using any editor and then make it executable. There are some points to watch here though.

German and English hyphenation

When formatting texts with the Documenter's Workbench (DWB) it is possible to enable language-specific hyphenation depending on whether the text is in German or in English. To set the language-specific hyphenation you use the macro

`.la` [*language*]

This has the following meaning:

`.la G`
German hyphenation

`.la E`
English hyphenation.

The default is German hyphenation, i.e. `.la G` or simply `.la` without parameters. The language setting is made directly at any position in the text file (generally the beginning) or in an include file (`.so`). In the case of "mixed language" documents with both German and English text, hyphenation can be switched as needed.

You should note that the default is for *no* hyphenation (except in words that contain the "-" or "\(em" [minus sign, dash; see `roff-charset(5)`] characters, where the word is split even if hyphenation is switched off). This means that hyphenation must always be switched on explicitly! To do this the macro

`.hy` [*N*]

is entered directly in the source file or in the macro package being used. In other words:

- The `.la` macro does not switch on the `.hy` macro implicitly.
- Hyphenation is only enabled in the text from the point where the `.hy` macro is first called, i.e. if it is called for the first time at the end of the file, no hyphenation will take place.

The *N* parameter of the `.hy` macro has the following meaning:

`1` Hyphenation with no restrictions

`2` No hyphenation with new page

`4` The last two letters of a word are not truncated

`8` The first two letters of a word are not truncated

`0` Hyphenation is switched off (= `.nh`)

The values 2, 4 and 8 can be combined freely, which means, for example, that `.hy 14` causes all three restrictions. `.hy` without parameters is identical with `.hy 1` (no restriction).

Controlling hyphenation

The `hyphen(1M)` command can be used to check that `nroff(1M)` hyphenation has been carried out properly. This command lists all words with hyphens at the line end in a formatted text. If you use `hyphen` as a filter, hyphenation in a file *textfile* can be controlled with the following command:

```
tbl textfile | neqn | nroff -man -Tlp | col -x | hyphen
```

Manipulating hyphenation

If the hyphenation is incorrect, there are two ways to manipulate it:

1) directly in the text by marking a mandatory hyphenation point using the `.hw` macro (for the general user)

2) by maintaining a (German) exception list (only for system administrators)

To mark a mandatory hyphenation point directly in the text, you need to enter the following character string

`\%`

(if another hyphenation point has not already been suggested with the `.hc` *c* macro).

The option of manually specifying a hyphenation point in a word is used particularly in texts with long words. The word "non-internationalized", for example, would then be edited as follows in the source file:

`non-in\%ter\%na\%tion\%al\%ized`.

A mandatory hyphenation point can also be specified with the

`.hw` *word*

macro. The hyphenation points are marked here with dashes, for example

`.hw dan-de-li-on`.

The *word* argument is 256 bytes long, and the maximum number of hyphenation points per word is 20.

Warning:

If a possible hyphenation point is given in a word, n/troff will *only* hyphenate at this point (and with the "- "sign). It is safer, therefore, to specify *all* mandatory hyphenation points (as in the above example).

Hyphenation can be suppressed in two ways: globally or simply for an individual word. This is usually done with the

`.nh` or `.hy 0`

macro.

To suppress hyphenation for a single word, either

- a `\%` is set at the start of the word, or
- the special characters "\(hy" (= -), "\(mi" (= –) or "\-" (= –) are inserted in a compound [see `roff-charset`(5)].

Here are two examples for suppressing hyphenation in a single word:

`\%automount` or `\%PPP\(hyPackage`.

A more elegant (but also more time-consuming) solution than hyphenating directly in the text is the possibility of specifying hyphenation points using an exception list. This procedure is worthwhile in the case of frequently used words which are systematically hyphenated incorrectly. It is worth noting, however, that while such an exception list only exists for German hyphenation, English words can of course also be entered in the list.

EDITING THE EXCEPTION LIST

The exception list generally comprises three individual files: `ausnahme.txt`, `ausnahme.dat` and `ausnahme.ind`. These files can be found in the `/usr/lib` directory. Entries are only made by the user in the `ausnahme.txt` input file, while the other two index files are generated and are not readable. You need superuser rights to make entries in the file. The editing of the exception list is somewhat complicated and therefore should be described in detail.

The words or parts of words in the exception list are marked twice:

a) by a *code letter* (type ID) at the start of the word, which identifies the entry as a word stem, beginning of a word, word form, foreign word etc.

b) by a *digit* between 1 and 4 within the entry, which specifies the actual hyphenation point

Code letters in the exception list

The list of *code letters* is now given below as well as their meanings.

`a` Normal exception word (word form only), i. e. the word only ever appears as an individual word, on its own.

Examples: wor-un-ter; wor-an (Entry: awor1un1ter; awor1an)

`b` Normal exception word (word form and beginning of word), i. e. the word appears on its own as well as at the beginning of compounds.

\Examples: Zeug-nis, Zeug-nis-ses, Zeug-nis-ar-ten (Entry: bzeug1nis)

`c` Normal exception word (beginning of word only), i. e. the word only ever appears at the beginning of a compound; in texts, a word marked in this way must always be followed another word part that contains at least two letters.

Examples: zi-vi-li-siert, zi-vi-li-sier-te (Entry: czi1vi1li)

`e` Proper noun (the "s" ending can also be appended to the word).

Examples: Ar-me-ni-en (Entry: ear1me1ni1en)

`f` Foreign word; the word can appear on its own, at the beginning of a compound and with the "s" ending.

Examples: Bo-nus, bo-nus-ar-tig (Entry: fbo1nus)

`g` Foreign word (beginning of word only), i.□e. the word is always joined with another word part (at least two letters) to form a compound.

Examples: Log-arith-mus, log-arith-misch (Entry: glog1arith)

`k` Customer entry; normal exception word (word form only), i.□e. the word only ever appears as an individual word, on its own (such as `a`).

`v` Verbs with "-en" stem.

Examples: gra-sen, gra-sen-de, Gras-an-satz (Entry: vgras)

`w` Verbs with "-eln" and "-ern" stem, including the two suffixes "el" and "er".

Examples: bet-teln, Bet-tel, Bett-ler, bet-teln-de (Entry: wbet1tel); zau-bern, Zau-be-rer, Zau-ber-kunst (Entry: wzau1ber)

`z` Word form from stem and ending, which is used in compounds.

Examples: Wis-sens, Wis-sens-art (Entry: zwis1sens)

All entries in the exception file must be written in lowercase.

Hyphenation point digits

The actual hyphenation points are indicated by the following *digits*:

`1` Normal hyphenation point.

Example: ain1itia1ti1ve (→ In-itia-ti-ve)

`2` Unaesthetic hyphenation point (should not be used!).

Unaesthetic hyphenation points includes those where the first or last two letters of a word are truncated. This type of hyphenation must be permitted explicitly using `.hy 1` (= unrestricted hyphenation). Tests have shown that words in the exception list that were marked with 2 for an unaesthetic hyphenation point, were not recognized subsequently by the system. If, however, they were marked with the digit 1, the hyphenation points were also recognized with unrestricted hyphenation as unaesthetic. The error can therefore be avoided easily, though the cause of the error is unknown.

`3` Hyphenation points when constants are added.

Example: bschif3fahrt (→ Schiff-fahrt)

`4` ck hyphenation.

Example: vfac4kel (→ Fak-kel)

Umlaut handling

With both code letters and digits being used to mark hyphenation points the entries in the exception list can look so strange that the user must first become familiar with reading them. In addition, German *umlauts* in the exception file must be entered using special non-alphabetical characters:

`{` for ä

Example: d{1cher (= Dä-cher)

`|` for ö

Example: d|r1fer (= Dör-fer)

`}` for ü

Example: f}n1fer (= Fün-fer)

`~` for ß

Example: au1~en (= au-ßen)

At first glance, all of these conventions (code letter, digits, umlauts, lowercase) make it difficult to maintain the

exception file. Moreover, there are also other rules that need to be observed, which affect the sort sequence of the entries.

Sort rules

It is critical that the entries in the exception list be *ordered* correctly so that it can function correctly. The entries must be in the correct *alphabetical* order, of course, so that the hyphenation program can actually find an entry in the exception list. For this reason, the basic word is taken *without* code letter and *without* hyphenation point digits. This is the basic rule.

Entries, in turn, may not only comprise complete words, but also word parts (word stem, beginning of word, word form). Contrary to these sort conventions, the long form must always come *before* the short form of a word or word part in the exception list. An example here is now given: There are four entries for the word part "gewinn" in the exception file. These four entries must be ordered as follows:

bge1win1ner□
zge1win3num1mern□
age1win3num1mer□
vge1winn

This example clarifies the first two sort rules:

- Long form before short form

  "gewinner" *before* "gewinn" and "gewinnummern" *before* "gewinnummer"
- Alphabetical order

  "gewinner" *before* "gewinnummern"

Other sort rules affect the ordering of entries with the alternative representations (non-alphabetical characters) of umlauts. You should note here that the non-alphabetical characters for umlauts must always be placed after the alphabetical characters, and that the actual umlauts are ordered alphabetically as normal without regard for their alternative representation, so for example ä before ö before ü. Here is another example from the exception file:

bge1fl}1gel	(= "geflügel")
bge1f{ng1nis	(= "gefängnis")
bge1f{~	(= "gefäß")
zge1f}hls	(= "gefühls")

The other sort rules are clarified with this example:

- Non-alphabetical characters (umlauts) after alphabetical

  bge1f{ng1nis ["gefängnis"] *after* bge1fl}1gel ["geflügel"], i.e. "f{" after "fl}"
- Umlauts in alphabetical order

  "gefängnis" alphabetically before "gefäß"; "gefäß" again alphabetically before "gefühls"

The next example from the exception file shows this again in more detail:

bbu~	(= "buß")
bb{c4ker	(= "bäcker")
zb{1ren	(= "bären")
zb|r1sen	(= "börsen")
cb|r1sia	(= "börsia")
cb|s1ar	(= "bösar")
zb}1cher	(= "bücher")
bb}1fett	(= "büfett")

bb}f1fe1lei	(= "büffelei")
wb}f1fel	(= "büffel")
wb}1gel	(= "bügel")
zb}h1nen	(= "bühnen")
bb}n1de1lei	(= "bündelei")
bb}n1del	(= "bündel")
vb}rg	(= "bürg")
bb}1ro	(= "büro")
zb}r1sten	(= "bürsten")
vb}rst	(= "bürst")
bb}t1tel	(= "büttel")
zb}t1ten	(= "bütten")
vb}~	(= "büß")

This example clarifies the previously mentioned sort rules:

- Long form before short form
  "büffelei" *before* "büffel" etc.
- Alphabetical ordering of word parts
  "börsen" *before* "börsia" etc.
- Umlauts in alphabetical order
  "bäcker" etc. *before* "börsen" etc. *before* "bücher" etc.
- Non-alphabetical characters (umlauts) after alphabetical
  bb{c4ker ("bäcker") *after* bbu~ ("buß") and not vice versa; vb}~ ("büß") *after* zb}t1ten ("bütten") and not vice versa.

A final sort rule can be derived from the preceding examples, with regard to non-alphabetical characters for umlauts. Since the non-alphabetical characters must always come after the alphabetical characters, entries of words or word parts that begin with an umlaut are also always ordered *after* the alphabetical entries (i.e. after the letter "z"). Here is a final example:

bzy1lin1der	(= "zylinder")
c{gyp	(= "ägyp")
c{ngst	(= "ängst")
b|l1ofen	(= "ölofen")
b}ber1ein	(= "überein")
b}ber1haupt	(= "überhaupt")

This example clarifies the following sort rules:

- Umlauts in alphabetical order
  "ägyp" etc. before "ölofen" etc. before "überein" etc.
- Non-alphabetical characters at the start of the entry after the letter "z"
  c{gyp ("ägyp") etc. after bzy1lin1der ("zylinder")

Aside from the rules for the correct ordering of entries, there are other points to look out for, which do not, however, interfere with the functioning of the exception list directly.

Notes

The following note applies in general:

- If certain words or word parts are entered in the exception list, this does *not* necessarily mean that the program would hyphenate them incorrectly without this entry. An exception may also be necessary in the list in order to hyphenate compounds with these entries correctly.
- The hyphenation program works on the basis of grammatical rules and probabilities, with the result that 100% accuracy cannot be achieved. Entries in the exception list have priority over the rules of the hyphenation algorithm.

Another note relates to possible error sources for the hyphenation program:

- Apart from foreign words, frequent sources of errors in German are compounds, although the hyphenation program is designed to deal with this. German words with so-called linking letters ("Fugen-Zeichen") are particularly problematic. Some examples of words hyphenated incorrectly if the exception list is not used are given below.

  Linking-s
  : Example: Abschnittsanzahl

    Hyphenation error without exception list: Ab-schnitt-san-zahl

  Linking-e
  : Example: Schweinefleisch

    Hyphenation error without exception list: Schwei-nef-leisch

  Linking -en
  : Example: Gruppenarbeiten

    Hyphenation error without exception list: Grup-pe-nar-bei-ten

  Compounds
  : Example: dialogorientiert

    Hyphenation error without exception list: dia-lo-go-rien-tiert

Yet another point to look at are special features of German hyphenation.

- Words that are *not* hyphenated in accordance with the other rules of German hyphenation must also be entered in the exception list. In particular, this affects the rule in German where "st" is not split. Exceptions to this rule, such as the words "Dienstag" (hyphenated as: Diens-tag), "Bistum" (hyphenated as: Bis-tum) or "Ostern" (not hyphenated) must be entered in the exception list.

  Warning:

  If a word is not to be hyphenated, it should be entered as follows: e.□g. `bostern` (for "Ostern") or `bangst` (for "Angst", "Angsthase" etc.) or `fguide` (for the English foreign word "guide"). A digit is thus not assigned for the hyphenation point, not even at the end of the word where a digit is generally not specified.

While it is possible to extend the number of entries in the exception list freely, it is better if it is restricted to a certain size.

- Rather than enter every word that has been hyphenated incorrectly in the exception list, it is worth considering if a word part would be sufficient. Thus, for example, the incorrectly hyphenated word "Abschnittsanzahl" could be entered as `aab1schnitts1an1zahl`. Even better, however, would be to enter it simply as the word part `zschnitts` since this covers various compounds at the same time, for example "Durchschnittsanalyse", "Querschnittsermittlung" etc. The *code letters* are provided in the exception file in order to support this effort at limiting the size.

The next note relates to the representation of umlauts in `troff`.

- Alternative umlaut representations in the source file, as are familiar to `troff(1M)` (\(^a = ä, \(^o = ö etc.), *cannot* be considered in this form in the exception list. Words with the `troff` alternative representation in the source code can only be split correctly if the umlaut is entered directly at the keyboard. If the split is still incorrect, the user has to specify the hyphenation point with `\%`, thus for example: `\(^uber\%haupt` or

```
.hw Ka-pi-tel-über-schrift.
```

Problems with hyphenation can always be anticipated if symbols such as ae, oe, ue etc. are used in the text instead of the real umlauts.

GENERATING THE NEW EXCEPTION LIST

After the user has made the new entries in the input file `ausnahme.txt`, a new, executable exception list must be generated. To do this use the command:

`genal` (without parameters)

This command reads the entries in the input file in succession, and creates a new executable exception list. The two files that make up the exception list are overwritten here (`ausnahme.dat` and `ausnahme.ind`). The command must be called in the directory where the input file and exception list are located.

During the formatting run, n/troff first searches for the exception list in the current directory and then under `/usr/lib`, if it is not available there.

PROBLEMS DESPITE THE NEW EXCEPTION LIST

After generating the new exception list, a small routine is available to check that words that were hyphenated incorrectly are now hyphenated correctly. The examples given above of incorrectly hyphenated words would have to be written here to a test file with the following contents:

```
 .hy 1              \" unrestricted hyphenation□
 .ll 2n             \" shortened line length□
Abschnittsanzahl□
                    \" real blank lines, no nofill mode!□
Schweinefleisch□
Gruppenarbeiten□
dialogorientiert
```

The command is simply as follows: `nroff testfile | hyphen`. If the entries have been recorded correctly in the exception list, the correct hyphenations can now be viewed on screen.

If the required result is not given however, and a word is still hyphenated incorrectly, the entry must be checked in the `ausnahme.txt` input file. From time to time, an error occurs where a non-applicable *code letter* is selected; more frequently, however, the *sort sequence* of the entries has not been observed strictly enough, with the result that the otherwise correct entry could not actually be found. The procedure must, in any case, be repeated until the result is correct.

In terms of hyphenation being correct, it is irrelevant whether `nroff` or `troff` formatting was used. .Sh "SEMANTIC PROBLEMS" Semantic conflicts may arise when hyphenating some German words, that also cannot be intercepted with the exception list. It depends on the context then whether, for example, the word "Baumast" is hyphenated as Baum-ast or as Bau-mast. In such a case, the user needs to insert an `\%` directly to specify what is meant: i.e. `Baum\%ast` or `Bau\%mast`.

SPELLING REFORM

The German hyphenation algorithm works in line with the hyphenation rules that are currently valid. The character string "ck" in a word is replaced with "k-k" when hyphenated (e.□g. Druk-ker) or a third consonant must be inserted (e.□g. Schiffahrt → Schiff-fahrt).

FILES

`/usr/lib/ausnahme.txt`
: Editable exception list

`/usr/lib/ausnahme.dat`
: Non-readable index file

`/usr/lib/ausnahme.ind`
: Non-readable index file

SEE ALSO

hyphen(1M), roff-charset(5).

System Administrator's Reference Manual

getdev(1M)

NAME

`getdev` - lists devices based on criteria

SYNOPSIS

`getdev` [`-ae`] [*criteria* [...]] [*device* [...]]

DESCRIPTION

`getdev` generates a list of devices that match certain criteria. The criteria include a list of attributes (given in expressions) and a list of devices. If no criteria are given, all devices are included in the list.

Devices must satisfy at least one of the criteria in the list unless the `-a` option is used. Then, only those devices which match all of the criteria in a list will be included.

Devices which are defined on the command line and which match the criteria are included in the generated list. However, if the `-e` flag is used, the list becomes a set of devices to be *excluded* from the list.

Criteria Expression Types

There are four possible expression types which the criteria specified in the *criteria* argument may follow:

attribute`=`*value*
: Selects all devices whose attribute *attribute* is defined and is equal to *value*.

attribute`!=`*value*
: Selects all devices whose attribute *attribute* is defined and does not equal *value*.

attribute`:*`
: Selects all devices which have the attribute *attribute* defined.

attribute`!:*`
: Selects all devices which do not have the attribute *attribute* defined.

See the `putdev`(1M) manual page for a complete listing and description of available attributes.

OPTIONS

`-a`
: Specifies that a device must match all criteria to be included in the list generated by this command. The flag has no effect if no criteria are defined.

`-e`
: Specifies that the list of devices which follows on the command line should be *excluded* from the list generated by this command. (Without the `-e` the named devices are *included* in the generated list.) The flag has no effect if no devices are defined.

criteria
: Defines criteria that a device must match to be included in the generated list. Should be given in expressions.

device
: Defines devices which should be included in the generated list. Can be the pathname of the device or the device alias.

DIAGNOSTICS

The command will exit with one of the following values:

`0` **=** Successful completion of the task.

`1` **=** Command syntax incorrect, invalid option used, or internal error occurred.

`2` **=** Device table could not be opened for reading.

FILES

`/etc/device.tab`

SEE ALSO

devattr(1M), getdgrp(1M), putdev(1M), putdgrp(1M).

System Administrator's Reference Manual

getdgrp(1M)

NAME

`getdgrp` - lists device groups which contain devices that matchcriteria

SYNOPSIS

`getdgrp` [`-ael`] [*criteria* [...]] [*dgroup* [...]]

DESCRIPTION

`getdgrp` generates a list of device groups that contain devices matching the given criteria. The criteria are given in the form of expressions.

criteria can be one expression or a list of expressions which a device must meet for its group to be included in the list generated by `getdgrp`. If no criteria are given, all device groups are included in the list.

Devices must satisfy at least one of the criteria in the list. However, the `-a` flag can be used to define that a "logical and" operation should be performed. Then, only those groups containing devices which match all of the criteria in a list will be included.

dgroup defines a set of device groups to be included in the list. Device groups that are defined and which contain devices matching the criteria are included. However, if the `-e` flag is used, this list defines a set of device groups to be excluded. When the `-e` option is used and criteria is also defined, the generated list will include device groups containing devices which match the criteria and are not in the command line list.

Criteria Expression Types

There are four possible expressions types:

attribute=*value*
: Selects all device groups with a member whose attribute *attribute* is defined and is equal to *value*.

attribute`!=`*value*
: Selects all device groups with a member whose attribute *attribute* is defined and does not equal *value*.

attribute`:*`
: Selects all device groups with a member which has the attribute *attribute* defined.

attribute`!:*`
: Selects all device groups with a member which does not have the attribute *attribute* defined.

See the `putdev(1M)` manual page for a complete listing and description of available attributes.

OPTIONS

`-a` Specifies that a device must match all criteria before a device group to which it belongs can be included in the list generated by this command. The flag has no effect if no criteria are defined.

`-e` Specifies that the list of device groups on the command line should be *excluded* from the list generated by this command. (Without the `-e` the named device groups are the only ones which can be *included* in the generated list.) The flag has no effect if no device groups are defined.

`-l` Specifies that all device groups (subject to the `-e` option and the *dgroup* list) should be listed even if they contain no valid device members. This option has no effect if *criteria* is specified on the command line.

criteria
: Defines criteria that a device must match before a device group to which it belongs can be included in the generated list.

dgroup
: Defines device groups which should be included in or excluded from the generated list.

DIAGNOSTICS

The command will exit with one of the following values:

`0` = Successful completion of the task.

`1` = Command syntax incorrect, invalid option used, or internal error occurred.

`2` = Device table or device group table could not be opened for reading.

FILES

`/etc/device.tab`

`/etc/dgroup.tab`

SEE ALSO

`devattr`(1M), `getdev`(1M), `putdev`(1M), `putdgrp`(1M).

System Administrator's Reference Manual

getemm(1M)

NAME

`getemm` - output error description(s)

SYNOPSIS

`getemm` *catalog*`:`*number*

`getemm –c` | `–n` *catalog*

`getemm –y`

DESCRIPTION

Depending on the language (`$LANG`), the `getemm` command outputs the error descriptions for the `/usr/lib/locale/$LANG/LC_MESSAGES/SIemM` directory.

If the `$LANG` language variable is not defined, the output occurs from the standard `/usr/lib/locale/C/LC_MESSAGES/SIemM` directory.

OPTIONS

`–c` Outputs all error descriptions for a catalog.

`–n` Outputs all error description numbers for a catalog.

`–y` Outputs all available catalog names.

catalog
: Name of a catalog.

number
: Number of an error message.

EXAMPLES

`getemm tca:100`
: Outputs the error description no. 100 for the `tca` catalog

`getemm –c "t*"`
: Outputs all error descriptions for the "`t*`" catalogs

`getemm –c tca`
: Outputs all error descriptions for the `tca` catalog

`getemm –n tca`
: Outputs all error description numbers for the `tca` catalog

`getemm –y`
: Outputs all catalog names

NOTES

Error messages produced by a debug kernel [see `utsdbg(5)`] are not recorded by `getemm`.

FILES

`/usr/lib/locale/$LANG/LC_MESSAGES/SIemM/*`

`/usr/lib/locale/C/LC_MESSAGES/SIemM/*`

SEE ALSO

`gettxt(1)`, `mkmsgs(1)`, `utsdbg(5)`.

System Administrator's Reference Manual

getty(1M)

NAME

`getty` - set terminal type, modes, speed, and line discipline

SYNOPSIS

`/usr/lib/saf/ttymon` [`–h`] [`–t` *timeout*] *line* [*speed* [*type* [*linedisc*]]]

`/usr/lib/saf/ttymon –c` *file*

DESCRIPTION

`getty` is a symbolic link to `/usr/lib/saf/ttymon`. It is included for compatibility with previous releases for the few applications that still call `getty` directly. `getty` can only be executed by the superuser, that is, by a process with the user ID `root`. Initially `getty` prints the login prompt, waits for the user's login name, and then invokes the `login` command. `getty` attempts to adapt the system to the terminal speed by using the options and arguments specified on the command line.

OPTIONS

line The name of a TTY line in `/dev` to which `getty` is to attach itself. `getty` uses this string as the name of a file in the `/dev` directory to open for reading and writing.

`–h` If the `–h` flag is not set, a hangup will be forced by setting the speed to zero before setting the speed to the default or specified speed.

`–t` *timeout*
specifies that `getty` should exit if the open on the line succeeds and no one types anything in *timeout* seconds.

speed
The *speed* argument is a label to a speed and TTY definition in the file `/etc/ttydefs`. This definition tells `getty` at what speed to run initially, what the initial TTY settings are, and what speed to try next, should the user indicate, by pressing the BREAK key, that the speed is inappropriate. The default *speed* is 300 baud.

type and *linedisc*
These options are obsolete and will be ignored.

`–c` *file*
The `–c` option is no longer supported. Instead use `sttydefs –l` to list the contents of the `/etc/ttydefs` file and perform a validity check on the file.

When given no optional arguments, `getty` specifies the following: The *speed* of the interface is set to 300 baud, either parity is allowed, new-line characters are converted to carriage return-line feed, and tab expansion is performed on the standard output. `getty` types the login prompt before reading the user's name a character at a time. If a null character (or framing error) is received, it is assumed to be the result of the user pressing the BREAK key. This will cause `getty` to attempt the next *speed* in the series. The series that `getty` tries is determined by what it finds in `/etc/ttydefs`.

FILES

`/etc/ttydefs`

SEE ALSO

`ct(1)`, `login(1)`, `sttydefs(1M)`, `ttymon(1M)`, `ioctl(2)`, `tty(7)`.

System Administrator's Reference Manual

getvol(1M)

NAME

`getvol -` verify device accessibility

SYNOPSIS

`getvol –n [ –l` *label* `] device`

`getvol [ –f|–F ] [ –wo ] [ –l` *label*`|–x` *label* `] device`

DESCRIPTION

`getvol` verifies that the specified device is accessible and that a volume of the appropriate medium has been inserted. The command is interactive and displays instructional prompts, describes errors, and shows required label information.

OPTIONS

`–n` Runs the command in non-interactive mode. The volume is assumed to be inserted upon command invocation.

`–l` Specifies that the label *label* must exist on the inserted volume (can be overridden by the `–o` option).

`–f` Formats the volume after insertion, using the format command defined for this device in the device table.

`–F` Formats the volume after insertion and places a file system on the device. Also uses the format command defined for this device in the device table.

`–w` Allows administrator to write a new label on the device. User is prompted to supply the label text. This option is ineffective if the `–n` option is enabled.

`–o` Allows the administrator to override a label check.

`–x` Specifies that the label *label* must exist on the device. This option should be used in place of the `–l` option when the label can only be verified by visual means. Use of the option causes a message to be displayed asking the administrator to visually verify that the label is indeed *label*.

device
Names the device which should be verified for accessibility.

DIAGNOSTICS

The command will exit with one of the following values:

`0` = Successful completion of the task.

`1` = Command syntax incorrect, invalid option used, or internal error occurred.

`3` = Device table could not be opened for reading.

NOTES

This command uses the device table to determine the characteristics of the device when performing the volume label checking.

FILES

`/etc/device.tab`

System Administrator's Reference Manual

grap(1M)

NAME

`grap` - pic preprocessor for drawing graphs

SYNOPSIS

`grap` *file* ...

DESCRIPTION

`grap` is a `pic`(1M) preprocessor for drawing graphs on a typesetter. Graphs are surrounded by the `troff` commands `.G1` and `.G2`. Data that is enclosed is scaled and plotted, with tick marks supplied automatically. Commands exist to modify the frame, add labels, override the default ticks, change the plotting style, define coordinate ranges and transformations, and include data from files. In addition, `grap` provides the same loops, conditionals and macro processing that `pic` does.

`frame ht` *e* `wid` *e* `top dotted` ...:
Set the frame around the graph to specified `ht` and `wid`; default is 2 by 3 (inches). The default mode applies to all four sides; `top`, `bot`, `left`, or `right` can be set to `dotted`, `dashed`, `invis`, or `solid` independently.

`label` *side* `"a label" "as a set of strings" adjust`:
Place label on specified side; default side is bottom. `adjust` is `up` (or `down`, `left`, `right`) *expr* to shift default position; `width` *expr* sets the width explicitly.

`ticks` *side* `in at` *optname expr*, *expr*, ...:
Put ticks on *side* at *expr*, ..., and label with *expr*. If any *expr* is followed by "...", label tick with "...", and turn off all automatic labels. If "..." contains `%f`'s, they will be interpreted as `printf` formatting instructions for the tick value. Ticks point `in` or `out` (default out). Tick iterator: instead of `at` ..., use `from` *expr* `to` *expr* `by` *o expr* where *o* is optionally `+-*/` for additive or multiplicative steps. `by` can be omitted, to give steps of size 1. If no ticks are requested, they are supplied automatically; suppress this with `ticks`
`off`. Automatic ticks normally leave a margin of 7% on each side; set this to anything by `margin =` *expr*.

`grid` *side linedesc* `at` *optname expr*, *expr*, ...:
Draw grids perpendicular to *side* in style *linedesc* at *expr*, ... Iterators and labels work as with ticks.

`coord` *optname* `x` *min*, *max* `y` *min*, max `log x log`
y: Set range of coords and optional log scaling on either or both. This overrides computation of data range. Default value of *optname* is current coordinate system (each `coord` defines a new coordinate system).

`plot "`*str*`" at` *point*; `"`*str*`" at` *point*:
Put *str* at *point*. Text position can be qualified with `rjust`, `ljust`, `above`, `below` after "...".

`line from` *point* `to` *point linedesc*:
Draw line from here to there. `arrow` works in place of `line`.

`next` *optname* `at` *point linedesc*:
Continue plot of data in *optname* to *point*; default is current.

`draw` *optname linedesc* ...:
Set mode for `next`: use this style from now on, and plot "..." at each point (if given).

`new` *optname linedesc* ...:
Set mode for `next`, but disconnect from previous.

A list of numbers *x y1 y2 y3* ... is treated as `plot bullet at`*x,y1*; `plot bullet at` *x,y2*; etc. or as `next`
`at` *x,y1* etc. if `draw` is specified. Abscissae of 1,2,3, ... are provided if there is only one input number per line.

A point *optname expr*, *expr* maps the point to the named coordinate system. A *linedesc* is one of `dot`, `dash`, `invis`, `solid` optionally followed by an expression.

`define` *name* { *whatever* }:
Define a macro. There are macros already defined for standard plotting symbols like `bullet`, `circle`, `star`, `plus`, etc. in `/usr/lib/dwb/grap.defines`, which is included if it exists.

var = *expr*:
Evaluate an expression. Operators are `=+-*` and `/`. Functions are `log` and `exp` (both base 10), `sin`, `cos`, `sqrt`; `rand` returns random number on [0,1]; `max(`*e*`,`*e*`)`, `min(`*e*`,`*e*`)`, `int(`*e*`)`.

`print` *expr*; `print` "...":
As a debugging aid, print *expr* or *string* on the standard error.

`copy "`*filename*`"`:
Include this file right here.

`copy thru` *macro*:
Pass rest of input (until `.G2`) through *macro*, treating each field (non-blank, or "...") as an argument. *macro* can be the name of a macro previously defined, or the body of one in place, like `/plot $1 at $2,$3/`.

`copy thru` *macro* `until "`*string*`"`:
Stop copy when input is *string* (left-justified).

`pic` *remainder of line*:
Copy to output with leading blanks removed.

`graph` *Name pic-position*:
Start a new frame, place it at specified position, e.g. `graph Thing2 with .sw at Thing1.se + (0.1,0)`. *Name* must be capitalized to keep `pic` happy.

`.`*anything at beginning of line*:
Copied verbatim.

`sh %` *anything* `%`:
Pass everything between the `%`'s to the shell; as with macros, `%` may be any character and *anything* may include newlines.

`#` *anything*:
A comment, which is discarded.

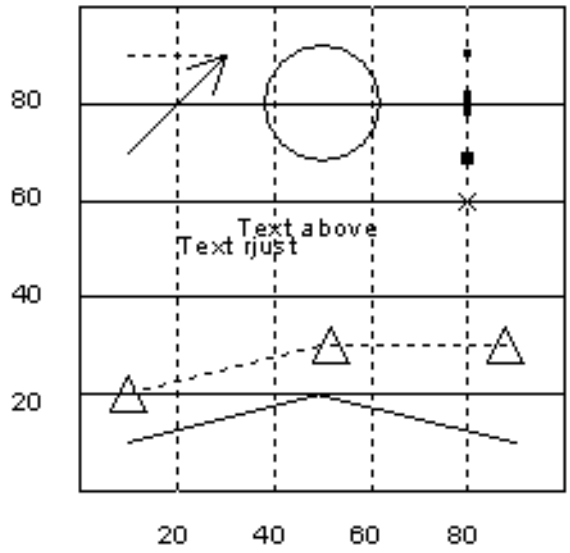
Order is mostly irrelevant; no category is mandatory. Any arguments on the `.G1` line are placed on the generated `.PS` line for `pic`.

EXAMPLES

```
.G1
frame ht 2 wid 2
coord x 0,100 y 0,100
grid dotted bot from 20 to 80 by 20
grid dotted left from 20 to 80 by 20
"Text above"   above at 50,50
"Text rjust "  rjust at 50,50
bullet at 80,90
vtick  at 80,80
box    at (80,70)
times  at 80,60
circle at 50,50
circle at 50,80 radius .25
line dashed from 10,90 to 30,90
arrow from 10,70 to 30,90
draw A solid
```

```
draw B dashed delta□
next A at 10,10□
next B at 10,20□
next A at 50,20□
next A at 90,10□
next B at 50,30□
next B at 90,30□
.G2
```

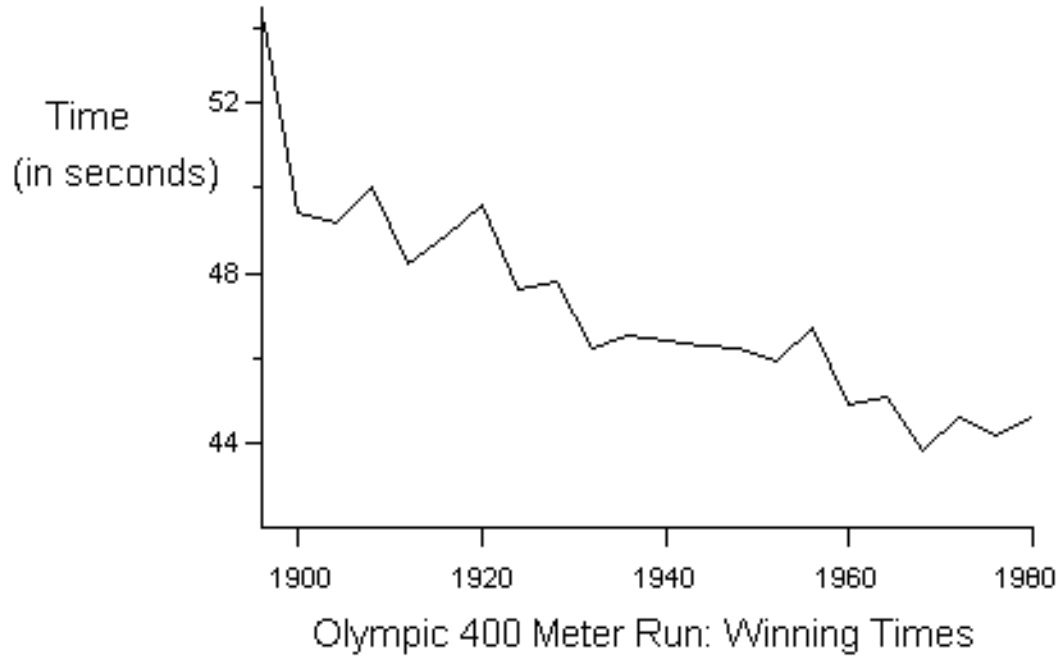
yields:




```
.G1□
frame invis ht 2 wid 3 left solid bot solid□
label left "Time" "(in seconds)" left .2□
label bot "Olympic 400 Meter Run: Winning Times"□
coord x 1894,1982 y 42,56□
ticks left out at 44 "44", 46, 48 "48", 50, 52 "52", 54□
ticks bot in from 1900 to 1980 by 20□
draw solid□
1896 54.2□
1900 49.4□
1904 49.2□
1908 50.0□
1912 48.2□
1920 49.6□
1924 47.6□
1928 47.8□
1932 46.2□
1936 46.5□
1948 46.2□
1952 45.9□
1956 46.7□
1960 44.9□
1964 45.1□
```

```
1968 43.8
1972 44.6
1976 44.2
1980 44.6
.G2
```

yields:



If you place the list of dates and winning times in a separate file called `400mpairs.d`, and then include this file with `copy`, the specification is reduced to the following lines:

```
.G1
frame invis ht 2 wid 3 left solid bot solid
label left "Time" "(in seconds)" left .2
label bot "Olympic 400 Meter Run: Winning Times"
coord x 1894,1982 y 42,56
ticks left out at 44 "44", 46, 48 "48", 50, 52 "52", 54
ticks bot in from 1900 to 1980 by 20
draw solid
copy "400mpairs.d"
.G2
```

FILES

`/usr/lib/dwb/grap.defines`

SEE ALSO

`pic`(1M) .

J. L. Bentley and B. W. Kernighan, *GRAP — A Language for Typesetting Graphs*, CSTR 114, 1984

System Administrator's Reference Manual

groupadd(1M)

NAME

`groupadd` - add (create) a new group definition on the system

SYNOPSIS

`groupadd` [–g *gid* [–o]] *group*

DESCRIPTION

The `groupadd` command creates a new group definition on the system by adding the appropriate entry to the `/etc/group` file.

OPTIONS

–g *gid*
: The group id for the new group. This group id must be a non-negative decimal integer below `MAXUID` as defined in the `<param.h>` header file. The group ID defaults to the next available (unique) number above the highest number currently assigned. For example, if groups 100, 105, and 200 are assigned as groups, the next default group number will be 201. (Group IDs from 0-99 are reserved.)

–o
: This option allows the *gid* to be duplicated (non-unique).

group
: A string with a maximum of 16 printable characters, which specifies the name of the new group. It may not include a colon (:) or newline (\n).

DIAGNOSTICS

The command exits with one of the following values:

`0` = Success.

`2` = Invalid command syntax. A usage message for the `groupadd` command is displayed.

`3` = An invalid argument was provided to an option.

`4` = *gid* is not unique (when –o option is not used).

`9` = *group* is not unique.

`10` = Cannot update the `/etc/group` file.

LOCALE

The `LC_MESSAGES` environment variable governs the language in which message texts are displayed.

If `LC_MESSAGES` is undefined or is defined as the null string, it defaults to the value of `LANG`. If `LANG` is likewise undefined or null, the system acts as if it were not internationalized.

If any of the locale variables has an invalid value, the system acts as if none of the variables was set.

The `LC_ALL` environment variable governs the entire locale. `LC_ALL` takes precedence over all the other environment variables that affect internationalization.

FILES

`/etc/group`

SEE ALSO

`users(1)`, `groupdel(1M)`, `groupmod(1M)`, `logins(1M)`, `useradd(1M)`, `userdel(1M)`, `usermod(1M)`.

System Administrator's Reference Manual

groupdel(1M)

NAME

`groupdel` - delete a group definition from the system

SYNOPSIS

`groupdel` *group*

DESCRIPTION

The `groupdel` command deletes a group definition from the system. It deletes the appropriate entry from the `/etc/group` file.

OPTIONS

group
A string of printable characters that specifies the group to be deleted.

DIAGNOSTICS

The command exits with one of the following values:

`0` = Success.

`2` = Invalid command syntax. A usage message for the `groupdel` command is displayed.

`6` = `group` does not exist.

`10` = Cannot update the `/etc/group` file.

LOCALE

The `LC_MESSAGES` environment variable governs the language in which message texts are displayed.

If `LC_MESSAGES` is undefined or is defined as the null string, it defaults to the value of `LANG`. If `LANG` is likewise undefined or null, the system acts as if it were not internationalized.

If any of the locale variables has an invalid value, the system acts as if none of the variables was set.

The `LC_ALL` environment variable governs the entire locale. `LC_ALL` takes precedence over all the other environment variables that affect internationalization.

FILES

`/etc/group`

SEE ALSO

`users(1)`, `groupadd(1M)`, `groupmod(1M)`, `logins(1M)`, `useradd(1M)`, `userdel(1M)`, `usermod(1M)`.

System Administrator's Reference Manual

groupmod(1M)

NAME

`groupmod` - modify a group definition on the system

SYNOPSIS

`groupmod` [`–g` *gid* [`–o`]] [`–n` *name*] *group*

DESCRIPTION

The `groupmod` command modifies the definition of the specified group by modifying the appropriate entry in the `/etc/group` file.

OPTIONS

`–g` *gid*
: The group id for the new group. This group id must be a non-negative decimal integer below MAXUID as defined in <`param.h`>. The group ID defaults to the next available (unique) number above 99. (Group IDs from 0-99 are reserved.)

`–o`
: This option allows the *gid* to be duplicated (non-unique).

`–n` *name*
: A string of printable characters that specifies a new name for the group. It may not include a colon (:) or newline (\n).

group
: The current name of the group to be modified.

DIAGNOSTICS

The command exits with one of the following values:

`0` = Success.

`2` = Invalid command syntax. A usage message for the `groupmod` command is displayed.

`3` = An invalid argument was provided to an option.

`4` = *gid* is not unique (when the `–o` option is not used).

`6` = *group* does not exist.

`9` = *name* already exists as a group name.

`10` = Cannot update the `/etc/group` file.

LOCALE

The `LC_MESSAGES` environment variable governs the language in which message texts are displayed.

If `LC_MESSAGES` is undefined or is defined as the null string, it defaults to the value of `LANG`. If `LANG` is likewise undefined or null, the system acts as if it were not internationalized.

If any of the locale variables has an invalid value, the system acts as if none of the variables was set.

The `LC_ALL` environment variable governs the entire locale. `LC_ALL` takes precedence over all the other environment variables that affect internationalization.

FILES

`/etc/group`

SEE ALSO

`users`(1), `groupadd`(1M), `groupdel`(1M), `logins`(1M), `useradd`(1M), `userdel`(1M), `usermod`(1M).

System Administrator's Reference Manual

grpck(1M)
BSD Compatibility Package

NAME

`grpck` - check group database entries

SYNOPSIS

`/usr/ucb/grpck` [*filename*]

DESCRIPTION

`grpck` checks that a file in `group` does not contain any errors; it checks the `/etc/group` file by default.

This command differs from `/usr/sbin/grpck` in its ability to correctly parse YP entries in `/etc/passwd`.

DIAGNOSTICS

`Too many/few fields`
: An entry in the group file does not have the proper number of fields.

`No group name`
: The group name field of an entry is empty.

`Bad character(s) in group name`
: The group name in an entry contains characters other than lowercase letters and digits.

`Invalid GID`
: The group ID field in an entry is not numeric or is greater than 65535.

`Null login name`
: A login name in the list of login names in an entry is null.

`Login name not found in password file`
: A login name in the list of login names in an entry is not in the password file.

FILES

`/etc/group`

SEE ALSO

`group(4)`, `passwd(4)`.

halt(1M) - iotail(1M)

System Administrator's Reference Manual

halt(1M)
BSD Compatibility Package

NAME

`halt` - stop the processor

SYNOPSIS

`/usr/ucb/halt` [`-lnqy`]

DESCRIPTION

`halt` writes out any information pending to the disks and then stops the processor.

`halt` normally logs the system shutdown to the system log daemon, `syslogd(1M)`, and places a shutdown record in the login accounting file `/var/adm/wtmp`. These actions are inhibited if the `-n` or `-q` options are set.

OPTIONS

`-l` No logging.

`-n` Prevent the `sync(1M)` before stopping.

`-q` Quick halt. No graceful shutdown is attempted.

`-y` Halt the system, even from a dialup terminal.

NOTE

This command is equivalent to `init 0`.

FILES

`/var/adm/wtmp`
Login accounting file

SEE ALSO

`init(1M)`, `reboot(1M)`, `shutdown(1M)`, `syslogd(1M)`.

System Administrator's Reference Manual

hyphen(1M)

NAME

`hyphen` - find hyphenated words

SYNOPSIS

`hyphen` [*file* ...]

DESCRIPTION

`hyphen` finds all the hyphenated words ending lines in *file*(*s*) and prints them on standard output. If no arguments are given or if `hyphen` encounters –, it uses standard input. Thus, `hyphen` may be used as a filter.

EXAMPLES

You would use the following command-line to proofread `nroff`'s hyphenation in *file*(*s*):

`mm` *file* ... | `hyphen`

NOTES

`hyphen` can't cope with hyphenated italic (or underlined words); it frequently will either miss them altogether or mishandle them. `hyphen` occasionally gets confused but with no ill effects other than spurious extra output.

SEE ALSO

`mm`(1M), `mmt`(1M), `nroff`(1M), `troff`(1M).

System Administrator's Reference Manual

idbuild(1M)

NAME

idbuild - build new system kernel

SYNOPSIS

/etc/conf/bin/idbuild [–M | –S | –L | –MK | –KM | –SK | –KS | –LK | –KL]

DESCRIPTION

This script builds a new Reliant UNIX system kernel using the current system configuration in /etc/conf. The script uses the shell variable $ROOT from the user's environment as its starting path. $ROOT should always be set to null, except for the special case of kernel development in a non-root source tree.

Building a new Reliant UNIX system image consists of generating new system configuration files, then link-editing the kernel and device driver object modules in the /etc/conf/pack.d object tree. To do this, idbuild first calls the internal command idnetconf, which selects according to the contents of net.default(4) and net.conf(4) the networking modules to be used to build either an "enhanced" or a "fallback" kernel [cf. protocol_stack(5)], then idconfig(1M) , which builds kernel configuration files, and idmkunix(1M) , which processes the configuration files and link-edits a new Reliant UNIX system image.

The system configuration files are built by processing the Master and System files representing device driver and tunable parameter specifications. For the Reliant UNIX system, the /etc/conf/cf.d/mdevice and /etc/conf/cf.d/mtune files represent the Master information while /etc/conf/cf.d/stune and the files specified in /etc/conf/sdevice.d/* represent the System information. The kernel also has file system type information defined in the files specified by /etc/conf/sfsys.d/* and /etc/conf/mfsys.d/*.

Once configured, a lock file is set in /etc/.new_unix causing the new kernel to replace /unix on the next system shutdown (i.e. entry to the init 0 state). The new kernel will be executed on next system reboot.

OPTIONS

–M Causes idbuild to produce a multiprocessor kernel.

–S Causes idbuild to produce a single processor kernel.

Warning: Only multiprocessor kernels can be created on an RM600.

–L Causes idbuild to produce a multiprocessor debug kernel.

–MK or –KM
Causes idbuild to retain configuration files conf.c, config.h, vector.c, fsconf.c, direct, and ifile files after a multiprocessor kernel build.

–SK or –KS
Causes idbuild to retain configuration files conf.c, config.h, vector.c, fsconf.c, direct, and ifile files after a single processor kernel build.

–LK or –KL
Causes idbuild to retain configuration files conf.c, config.h, vector.c, fsconf.c, direct, and ifile files after a debug kernel build.

Only one of these options can be specified at a time.

DIAGNOSTICS

Since idbuild calls other system commands to perform system reconfiguration and link editing, it will report all errors encountered by those commands, then clean up intermediate files created in the process. In general, the exit value 1 indicates an error was encountered by idbuild.

The errors encountered fall into the following categories:

- Master file error messages.
- System file error messages.
- Tunable file error messages.
- Compiler and Link editor error messages.

All error messages are designed to be self-explanatory.

NOTES

The command `idbuild` can only be called in multiuser mode on an RM600.

SEE ALSO

`uname(1)`, `idinstall(1M)`, `idtune(1M)`, `mdevice(4)`, `mfsys(4)`, `mtune(4)`, `net.conf(4)`, `sdevice(4)`, `sfsys(4)`, `stune(4)`, `protocol_stack(5)`.

System Administrator's Reference Manual

idcheck(1M)

NAME

`idcheck -` returns selected information

SYNOPSIS

`/etc/conf/bin/idcheck`

DESCRIPTION

This command returns selected information about the system configuration. It is useful in add-on device Driver Software Package (DSP) installation scripts to determine if a particular device driver has already been installed. The forms are:

`idcheck –p` *device-name* [`–i` *dir*]

`idcheck –y` *device-name* [`–i` *dir*]

This command scans the System and Master modules and returns:

`100` if an error occurs

`0` if no conflict exists

`a positive number greater than 0 and less than 100`
if a conflict exists

OPTIONS

`–p` *device-name*
This option checks for the existence of four different components of the DSP. The exit code is the addition of the return codes from the four checks.

Add 1 to the exit code if the DSP directory under `/etc/conf/pack.d` exists.

Add 2 to the exit code if the Master module has been installed.

Add 4 to the exit code if the System module has been installed.

Add 8 to the exit code if the Kernel was built with the System module.

Add 16 to the exit code if a `Driver.o`, or a `SP_Driver.o` and `MP_Driver.o`, are part of the DSP (vs. a `stubs.c` file).

`–y` *device-name*
This option checks to see if the DSP will be configured into the next kernel to be built.

`–i` *dir*
Specifies the directory in which the files `sdevice` and `mdevice` reside. The default directory is `/etc/conf/cf.d`.

DIAGNOSTICS

There are no error messages or checks for valid arguments to options. `idcheck` interprets these arguments using the rules of `scanf(3S)` and queries the `sdevice` and `mdevice` files. For example, if a letter is used in the place of a digit, `scanf(3S)` will translate the letter to 0. `idcheck` will then use this value in its query.

SEE ALSO

`idinstall(1M)`, `mdevice(4)`, `sdevice(4)`.

System Administrator's Reference Manual

idconfig(1M)

NAME

`idconfig -` produce a new kernel configuration

SYNOPSIS

`/etc/conf/bin/idconfig`

DESCRIPTION

The `idconfig` command takes as its input a collection of files specifying the configuration of the next Reliant UNIX system to be built. A collection of output files for use by `idmkunix(1M)` is produced.

The input files expected by `idconfig` are as follows:

`mdevice`
– Master device specifications

`sdevice`
– System device specifications

`mtune`
– Master parameter specifications

`stune`
– System parameter specifications

`mfsys`
– File system type master data

`sfsys`
– File system type system data

`sassign`
– Device Assignment File

`sema` – STREAMS multiprocessor configuration file

The output files produced by `idconfig` are as follows:

`conf.c`
– Kernel data structures and function definitions

`config.h`
– Kernel parameter and device definitions

`direct`
– Listing of all driver components included in the build

`fsconf.c`
– File system type configuration data

OPTIONS

`–o` *directory*
Output files will be created in the directory specified rather than `/etc/conf/cf.d`.

`–i` *directory*
Input files that normally reside in `/etc/conf/cf.d` can be found in the directory specified.

`–r` *directory*
The directory specified will be used as the ID "root" directory rather than `/etc/conf`.

`–d` *file*
Use *file* name rather than `sdevice` for input.

`–t` *file*
Use *file* name rather than `stune` for input.

–T *file*
Use *file* name rather than `mtune` for input.

–a *file*
Use *file* name rather than `sassign` for input.

–c *file*
Redirect `conf.c` output to *file* name.

–h *file*
Redirect `config.h` output to *file* name.

–p *file*
Redirect `direct` output to *file* name.

–s Suppress searching driver packages.

–M Configure a multiprocessor kernel; default is to configure a single processor kernel.

–# Print debugging information.

This version of Reliant UNIX does not support multiple major numbers for drivers. `idconfig` generates additional constants (via defines) in the `config.h` file so that they can be used by the driver (as they will get referenced in the `space.c` file) to generate appropriate data structures. The information provided by these constants is how many major numbers were assigned to the device and what are their values. The names of the constants are as follows:

prfx`_CMAJOR_`*x*
prfx`_BMAJOR_`*x*

where *prfx* stands for device prefix. The *x* stands for the list subscript, starting with subscript `0`.

In addition, the configuration file `conf.c` that initializes `bdevsw[]` and `cdevsw[]` tables will also add entries for each of the major numbers and, as such, the same driver entry points will be repeated for each one of the entries.

DIAGNOSTICS

An exit value of zero indicates success. If an error was encountered, `idconfig` will exit with a non-zero value and report an error message. All error messages are designed to be self-explanatory.

SEE ALSO

`idbuild`(1M), `idinstall`(1M), `idmkunix`(1M), `mdevice`(4), `mtune`(4), `sdevice`(4), `stune`(4).

System Administrator's Reference Manual

idinstall(1M)

NAME

`idinstall -` add, delete, update, or get device driver configurationdata

SYNOPSIS

`/etc/conf/bin/idinstall –[ adug ] [ –e ] [ –k ] –[ msoptnirhcl ]` *dev_name*

DESCRIPTION

The `idinstall` command is called by a Driver Software Package (DSP) Install script or Remove script to Add (`–a`), Delete (`–d`), Update (`–u`), or Get (`–g`) device driver configuration data. `idinstall` expects to find driver component files in the current directory. When components are installed or updated, they are moved or appended to files in the `/etc/conf` directory and then deleted from the current directory unless the `–k` flag is used.

OPTIONS

Action Specifiers:

`–a` Add the DSP components

`–d` Remove the DSP components

`–u` Update the DSP components

`–g` Get the DSP components (print to stdout, except Master)

Component Specifiers: (*)

`–m` Master component

`–s` System component

`–o` Driver object and optional support library components

`–p` Space.c component

`–t` Stubs.c component

`–n` Node (special file) component

`–i` Inittab component

`–r` Device Initialization (rc) component

`–h` Device shutdown (sd) component

`–c` Mfsys component: file system type config (Master) data

`–l` Sfsys component: file system type local (System) data

(*) If no component is specified, the default is all except for the `–g` option where a single component must be specified explicitly.

Miscellaneous:

`–e` Disable free disk space check

`–k` Keep files (do not remove from current directory) on add or update

Driver objects (`Driver.o`, `SP_Driver.o`, `MP_Driver.o`) and support libraries (*dev_name*`.a`, `SP_`*dev_name*`.a`, `MP_`*dev_name*`.a`) have an optional prefix which indicates support for single processor or multiprocessor kernels. A prefix of `SP_` indicates that the object or library is used for the configuration of single processor kernels. Similarly, a prefix of `MP_` is used for multiprocessor kernels. A driver or library without a prefix may be used for the configuration of either type of kernel.

In the simplest case of installing a new DSP, the command syntax used by the DSP's Install script should be `idinstall –a`*dev_name*. In this case the command will require and install the driver objects, `Master` and

`System` entry, and optionally install the support libraries, `Space.c`, `Stubs.c`, `Node`, `Init`, `Rc`, `Shutdown`, `Mfsys`, and `Sfsys` components if those modules are present in the current directory.

The Driver objects, support libraries, `Space.c` , and `Stubs.c` files are moved to a directory in `/etc/conf/pack.d`. The *dev_name* is passed as an argument, which is used as the directory name. The remaining components are stored in the corresponding directories under `/etc/conf` in a file whose name is *dev_name* For example, the `Node` file would be moved to `/etc/conf/node.d/dev_name`.

The `idinstall –m` usage provides an interface to the `idmaster` command which will add, delete, and update `mdevice` file entries using a `Master` file from the local directory. An interface is provided here so that driver writers have a consistent interface to install any DSP component.

As stated above, driver writers will generally use only the `idinstall –a` *dev_name* form of the command. Other options of `idinstall` are provided to allow an Update DSP (i.e. one that replaces an existing device driver component) to be installed, and to support installation of multiple controller boards of the same type.

If the call to `idinstall` uses the `–u` (update) option, it will:

- overlay the files of the old DSP with the files of the new DSP.
- invoke the `idmaster` command with the "update" option if a `Master` module is part of the new DSP.

`idinstall` also does a verification that enough free disk space is available to start the reconfiguration process. This is done by calling the `idspace` command. `idinstall` will fail if insufficient space exists, and exit with a non-zero return code. The `–e` option bypasses this check.

`idinstall` makes a record of the last device installed in a file (`/etc/.last_dev_add`), and saves all removed files from the last delete operation in a directory (`/etc/.last_dev_del`).

DIAGNOSTICS

An exit value of zero indicates success. If an error was encountered, `idinstall` will exit with a non-zero value, and report an error message. All error messages are designed to be self-explanatory. Typical error message that can be generated by `idinstall` are as follows:

```
Device package already exists.
Cannot make the driver package directory.
Cannot remove driver package directory.
Local directory does not contain a Driver object (Driver.o) file.
Local directory does not contain a Master file.
Local directory does not contain a System file.
Cannot remove driver entry.
```

SEE ALSO

`idspace(1M)`, `idcheck(1M)`, `mdevice(4)`, `sdevice(4)`.

System Administrator's Reference Manual

idinstallf(1M)

NAME

`idinstallf -` add a driver component to the software installationdatabase

SYNOPSIS

`idinstallf [ –u | –f ] [ –c` *class* `]` *pkginst module*

DESCRIPTION

`idinstallf` is called by the driver software package (DSP) installation script. Components of a DSP are typically installed by the `idinstall` command. `idinstallf` examines the `/etc/conf` directory for driver component files and adds these files to the installation database with type `f` and class *class* (default: `none`). The files `/etc/conf/cf.d/mdevice` and `sdevice` are added to the installation database with type `v` instead of `f`. Directories are stored as implicit directories with the current attribute values of the pathname.

OPTIONS

`–f` Indicates that the installation is complete. This option has the same effect as the `–f` option of the `installf` command.

`–u` Displays a help text.

`–c` *class*
Class to which installed objects should be assigned. The default class is `none`.

pkginst
Name of the package instance with which the driver component pathnames should be associated.

module
Name used as a directory name under `/etc/conf/pack.d` and as a file name under the corresponding `/etc/conf` directories.

`idinstallf` has the same effect as the `installf` commando applied to each of the available driver components.

There is no `idremovef` command corresponding to `removef`.

SEE ALSO

`idinstall`(1M), `installf`(1M), `removef`(1M), `mdevice`(4), `sdevice`(4).

System Administrator's Reference Manual

idintconf(1M)
RM400 only

NAME

`idintconf -` process the idevice file

SYNOPSIS

`idintconf` [`–d` *ifile*] [`–s` *sdevice*] [`–r` *directory*]

DESCRIPTION

`idintconf` takes as input an `idevice` file and generates a `intconf.c` file which is compiled and linked at kernel generation time. This file contains data structures which allow the kernel to handle the allocation of interrupts on the EISA and ISA bus.

OPTIONS

`–d` *ifile*
: This option allows the use of a different input file to the default `idevice` file.

`–s` *sdevice*
: `idintconf` uses the `sdevice` file to determine which entries in the `idevice` file are valid and to differentiate between two or more similar entries. The `sdevice` file which is used by `idintconf` can be changed by using this option.

`–r` *directory*
: The directory specified will be used as the ID "root" directory rather than `/etc/conf`.

FILES

`idevice`
`sdevice`
`intconf.c`

System Administrator's Reference Manual

idmknod(1M)

NAME

`idmknod` - removes nodes and reads specifications of nodes

SYNOPSIS

`/etc/conf/bin/idmknod`

DESCRIPTION

This command performs the following functions:

Removes the nodes for non-required devices (those that do not have an "r" in field 3 of the device's `mdevice` entry) from `/dev`. Ordinary files will not be removed. If the `/dev` directory contains subdirectories, those subdirectories will be checked and nodes found for non-required devices will be removed as well. If empty subdirectories result due to the removal of nodes, the subdirectories are then removed.

Reads the specifications of nodes given in the files contained in `/etc/conf/node.d` and installs these nodes in `/dev`. If the node specification defines a path containing subdirectories, the subdirectories will be made automatically.

Returns 0 on success and a positive number on error.

The files in `/etc/conf/node.d` are copies of the `Node` modules installed by device Driver Software Packages (DSP). There is at most one file per DSP. Each file contains one line for each node that is to be installed. The format of each line is:

- Name of device entry (field 1) in the `mdevice` file (The `mdevice` entry will be the line installed by the DSP from its `Master` module). This field must be from 1 to 8 characters in length. The first character must be a letter. The others may be letters, digits, or underscores.
- Name of node to be inserted in `/dev`. The first character must be a letter. The others may be letters, digits, or underscores. This field can be a path relative to `/dev`, and `idmknod` will create subdirectories as needed.
- The character `b` or `c`. A `b` indicates that the node is a "block" type device and `c` indicates "character" type device.
- Minor device number. This value normally lies between 0 and 255. The minor number may be extended to accommodate for the presence of a range of major numbers. The minor number integer divided by 255 results in a value which is added to the major number. This addition gives a major number which is an offset to the device base major number. If this field is a non-numeric, it is assumed to be a request for a streams clone device node, and `idmknod` will set the minor number to the value of the major number of the device specified.

Some example node file entries are as follows:

`asy tty00 c 1`
: makes `/dev/tty00` for device "asy" using minor device 1.

`qt rmt/c0s0 c 4`
: makes `/dev/rmt/c0s0` for device "qt" using minor device 4.

`clone net/nau/clone c nau`
: makes `/dev/net/nau/clone` for device "clone". The minor device number is set to the major device number of device "nau".

OPTIONS

`-o` *directory*
: Nodes will be installed in the directory specified rather than `/dev`.

`-i` *directory*
: The `mdevice` and `sdevice` files that normally reside in `/etc/conf/cf.d` can be found in the

directory specified.

`-e` *directory*
The `Node` modules that normally reside in `/etc/conf/node.d` can be found in the directory specified.

`-s` Suppress removing nodes (just add new nodes).

ERROR MESSAGES

An exit value of zero indicates success. If an error was encountered due to a syntax or format error in a `node` entry, an advisory message will be printed to *stdout* and the command will continue. If a serious error is encountered (i.e. a required file cannot be found), `idmknod` will exit with a non-zero value and report an error message. All error messages are designed to be self-explanatory.

SEE ALSO

`idinstall`(1M), `mdevice`(4), `sdevice`(4).

System Administrator's Reference Manual

idmkunix(1M)

NAME

`idmkunix` - build new system kernel

SYNOPSIS

`/etc/conf/bin/idmkunix`

DESCRIPTION

The `idmkunix` command creates a bootable Reliant UNIX operating system kernel in the directory `/etc/conf/cf.d`. The component kernel "core" files and device driver object files contained in subdirectories of `/etc/conf/pack.d` are used as input along with device and parameter definition files produced by `idconfig`. In brief, the required input files are as follows:

`/etc/conf/cf.d/conf.c`	–	Kernel data structures and function definitions
`/etc/conf/cf.d/config.h`	–	Kernel parameter and device definitions
`/etc/conf/cf.d/direct`	–	Listing of all driver components included in the build
`/etc/conf/cf.d/fsconf.c`	–	File system type configuration data
`/etc/conf/pack.d/*/*Driver.o`	–	Component kernel object files
`/etc/conf/pack.d/*/*.a`	–	Component kernel support libraries
`/etc/conf/pack.d/*/space.c`	–	Component kernel space allocation files
`/etc/conf/pack.d/*/stubs.c`	–	Component kernel stubs files

OPTIONS

`-o` *directory*
: The file *unix* be created in the directory specified rather than `/etc/conf/cf.d`.

`-i` *directory*
: Input files that normally reside in `/etc/conf/cf.d` can be found in the directory specified.

`-r` *directory*
: The directory specified will be used as the ID "root" directory rather than `/etc/conf`.

`-M`
: Creates a multiprocessor kernel. The default is a single processor kernel.

`-#`
: Print debugging information.

DIAGNOSTICS

An exit value of zero indicates success. If an error was encountered, `idmkunix` will exit with a non-zero value and report an error message. All error messages are designed to be self-explanatory.

SEE ALSO

`idbuild`(1M), `idconfig`(1M), `idinstall`(1M), `mdevice`(4), `mtune`(4), `sdevice`(4), `stune`(4).

System Administrator's Reference Manual

idspace(1M)

NAME

idspace - investigate free space

SYNOPSIS

/etc/conf/bin/idspace [–i *inodes*] [–r *blocks*] [–u *blocks*] [–t *blocks*]

DESCRIPTION

This command investigates free space in /, /usr, and /tmp file systems to determine whether sufficient disk blocks and inodes exist in each of potentially 3 file systems. The default tests that idspace performs are as follows:

1) Verify that the root file system (/) has 400 blocks more than the size of the current /unix. This verifies that a device driver being added to the current /unix can be built and placed in the root directory. A check is also made to ensure that 100 inodes exist in the root directory.

2) Determine whether a /usr file system exists. If it does exist, a test is made that 400 free blocks and 100 inodes are available in that file system. If the file system does not exist, idspace does not complain since files created in /usr by the reconfiguration process will be created in the root file system and space requirements are covered by the test in 1) above.

3) Determine whether a /tmp file system exists. If it does exist, a test is made that 400 free blocks and 100 inodes are available in that file system. If the file system does not exist, idspace does not complain since files created in /tmp by the reconfiguration process will be created in the root file system and space requirements are covered by the test in 1) above.

OPTIONS

–i *inodes*
: This option overrides the default test for 100 inode in all of the idspace checks.

–r *blocks*
: This option overrides the default test for /unix size + 400 blocks when checking the root (/) file system. When the –r option is used, the /usr and /tmp file systems are not tested unless explicitly specified.

–u *blocks*
: This option overrides the default test for 400 blocks when checking the /usr file system. When the –u option is used, the root (/) and /tmp file systems are not tested unless explicitly specified. If /usr is not a separate file system, an error is reported.

–t *blocks*
: This option overrides the default test for 400 blocks when checking the /tmp file system. When the –t option is used, the root (/) and /usr file systems are not tested unless explicitly specified. If /tmp is not a separate file system, an error is reported.

DIAGNOSTICS

An exit value of zero indicates success. If insufficient space exists in a file system or an error was encountered due to a syntax or format error, idspace will report a message. All error messages are designed to be self-explanatory. The specific exit values are as follows:

0 = Success.

1 = Command syntax error, or needed file does not exist.

2 = File system has insufficient space or inodes.

3 = Requested file system does not exist (–u and –t options only).

SEE ALSO

idbuild(1M), idinstall(1M).

System Administrator's Reference Manual

idtune(1M)

NAME

idtune - attempts to set the value of a tunable parameter

SYNOPSIS

/etc/conf/bin/idtune [−f] [−m] *name value*

DESCRIPTION

This script attempts to set the value of a tunable parameter. The tunable parameter to be changed is indicated by *name*. The desired value for the tunable parameter is *value*.

If there is already a value for this parameter (in the stune file), the user will normally be asked to confirm the change with the following message:

Tunable Parameter *name* is currently set to *old_value*.
Is it OK to change it to *value*? (y/n)

If the user answers y, the change will be made. Otherwise, the tunable parameter will not be changed, and the following message will be displayed:

name left at *old_value*.

However, if the −f (force) option is used, the change will always be made and no messages will ever be issued.

If the −m (minimum) option is used and there is an existing value which is greater than the desired value, no change will be made and no message will be issued.

If either the existing value or the desired value is a formula, the −m option will be ignored.

Using −f and −m options:

If system tunable parameters are being modified as part of a device driver or application add-on package, it may not be desirable to prompt the user with the above question. The install script for the add-on package may chose to override the existing value using the −f or −m options. However, care must be taken not to invalidate a tunable parameter modified earlier by the user or another add-on package.

In order for the change in parameter to become effective, the Reliant UNIX system kernel must be rebuilt and the system rebooted.

The sysctl interface provides another, more comfortable mechanism to influence the behavior of an "enhanced" system [cf. protocol_stack(5)] as is provided by the idtune tool. sysctl(1M) can be used to modify a running system; there is no need to link and reboot a new kernel for a change to be effective, but this functionality is essentially limited to networking tunables.

DIAGNOSTICS

The exit status will be non-zero if errors are encountered.

SEE ALSO

idbuild(1M), sysctl(1M), mtune(4), stune(4), protocol_stack(5).

System Administrator's Reference Manual

infocmp(1M)

NAME

infocmp - compare or print out terminfo descriptions

SYNOPSIS

infocmp [–d] [–c] [–n] [–I] [–L] [–C] [–r] [–u] [–s d|i|l|c] [–v] [–V] [–1] [–w *width*] [–A *directory*] [–B *directory*] [*termname ...*]

DESCRIPTION

infocmp can be used to compare a binary terminfo entry with other terminfo entries, rewrite a terminfo description to take advantage of the use= terminfo field, or print out a terminfo description from the binary file (term) in a variety of formats. In all cases, the boolean fields will be printed first, followed by the numeric fields, followed by the string fields.

Default Options

If no options are specified and zero or one *termnames* are specified, the –I option is assumed. If more than one *termname* is specified, the –d option is assumed.

Comparison Options [–d] [–c] [–n]

infocmp compares the terminfo description of the first terminal *termname* with each of the descriptions given by the entries for the other terminal's *termnames*. If a capability is defined for only one of the terminals, the value returned will depend on the type of the capability: F for boolean variables, –1 for integer variables, and NULL for string variables.

–d Produces a list of each capability that is different between two entries. This option is useful to show the difference between two entries, created by different people, for the same or similar terminals.

–c Produces a list of each capability that is common between two entries. Capabilities that are not set are ignored. This option can be used as a quick check to see if the –u option is worth using.

–n Produces a list of each capability that is in neither entry. If no *termnames* are given, the environment variable TERM will be used for both of the *termnames*. This can be used as a quick check to see if anything was left out of a description.

Source Listing Options [–I] [–L] [–C] [–r]

The –I, –L, and –C options will produce a source listing for each terminal named.

–I Use the terminfo names

–L Use the long C variable name listed in <term.h>

–C Use the termcap names

–r When using –C, put out all capabilities in termcap form

If no *termnames* are given, the environment variable TERM will be used for the terminal name.

The source produced by the –C option may be used directly as a termcap entry, but not all of the parameterized strings may be changed to the termcap format. infocmp will attempt to convert most of the parameterized information, but anything not converted will be plainly marked in the output and commented out. These should be edited by hand.

All padding information for strings will be collected together and placed at the beginning of the string where termcap expects it. Mandatory padding (padding information with a trailing "/") will become optional.

All termcap variables no longer supported by terminfo, but which are derivable from other terminfo variables, will be output. Not all terminfo capabilities will be translated; only those variables that were part of termcap will normally be output. Specifying the –r option will lift this restriction, allowing all capabilities to be output in termcap form.

Note that because padding is collected to the beginning of the capability, not all capabilities are output. Mandatory padding is not supported. Because `termcap` strings are not as flexible, it is not always possible to convert a `terminfo` string capability into an equivalent `termcap` format. A subsequent conversion of the `termcap` file back into `terminfo` format will not necessarily reproduce the original `terminfo` source.

Some common `terminfo` parameter sequences, their `termcap` equivalents, and some terminal types which commonly have such sequences, are:

terminfo	termcap	Representative Terminals
%p1%c	%.	adm
%p1%d	%d	hp, ANSI standard, vt100
%p1%'x'%+%c	%+x	concept
%i	%i	ANSI standard, vt100
%p1%?%'x'%>%t%p1%'y'%+%;	%>xy	concept
%p2 is printed before %p1	%r	hp

Use= Option [–u]

–u Produces a `terminfo` source description of the first terminal *termname* which is relative to the sum of the descriptions given by the entries for the other terminals *termnames*. It does this by analyzing the differences between the first *termname* and the other *termnames* and producing a description with `use=` fields for the other terminals. In this manner, it is possible to retrofit generic terminfo entries into a terminal's description. Or, if two similar terminals exist, but were coded at different times or by different people so that each description is a full description, using `infocmp` will show what can be done to change one description to be relative to the other.

A capability will get printed with an 'at' sign (`@`) if it no longer exists in the first *termname*, but one of the other *termname* entries contains a value for it. A capability's value gets printed if the value in the first *termname* is not found in any of the other *termname* entries, or if the first of the other *termname* entries that has this capability gives a different value for the capability than that in the first *termname*.

The order of the other *termname* entries is significant. Since the terminfo compiler `tic` does a left-to-right scan of the capabilities, specifying two `use=` entries that contain differing entries for the same capabilities will produce different results depending on the order that the entries are given in. `infocmp` will flag any such inconsistencies between the other *termname* entries as they are found.

Alternatively, specifying a capability after a `use=` entry that contains that capability will cause the second specification to be ignored. Using `infocmp` to recreate a description can be a useful check to make sure that everything was specified correctly in the original source description.

Another error that does not cause incorrect compiled files, but will slow down the compilation time, is specifying extra `use=` fields that are superfluous. `infocmp` will flag any other *termname* `use=` fields that were not needed.

Other Options [–s d|i|l|c] [–v] [–V] [–1] [–w width]

–s sorts the fields within each type according to the argument below:

- d Leave fields in the order that they are stored in the `terminfo` database.
- i Sort by `terminfo` name.
- l Sort by the long C variable name.
- c Sort by the `termcap` name.

If the –s option is not given, the fields printed out will be sorted alphabetically by the `terminfo` name within each type, except in the case of the –C or the –L options, which cause the sorting to be done by the `termcap` name or the long C variable name, respectively.

–v Prints out tracing information on standard error as the program runs.

`-V` Prints out the version of the program in use on standard error and exit.

`-1` Causes the fields to be printed out one to a line. Otherwise, the fields will be printed several to a line to a maximum width of 60 characters.

`-w` Changes the output to *width* characters.

Changing Databases [-A directory] [-B directory]

The location of the compiled `terminfo` database is taken from the environment variable `TERMINFO`. If the variable is not defined, or the terminal is not found in that location, the system `terminfo` database, usually in `/usr/share/lib/terminfo`, will be used. The options `-A` and `-B` may be used to override this location. The `-A` option will set `TERMINFO` for the first *termname* and the `-B` option will set `TERMINFO` for the other *termnames*. With this, it is possible to compare descriptions for a terminal with the same name located in two different databases. This is useful for comparing descriptions for the same terminal created by different people.

LOCALE

The `LC_MESSAGES` environment variable governs the language in which message texts are displayed.

If `LC_MESSAGES` is undefined or is defined as the null string, it defaults to the value of `LANG`. If `LANG` is likewise undefined or null, the system acts as if it were not internationalized.

If any of the locale variables has an invalid value, the system acts as if none of the variables was set.

The `LC_ALL` environment variable governs the entire locale. `LC_ALL` takes precedence over all the other environment variables that affect internationalization.

FILES

`/usr/share/lib/terminfo/?/*`
Compiled terminal description database

SEE ALSO

`captoinfo`(1M), `tic`(1M), `curses`(3X), `terminfo`(4).

System Administrator's Reference Manual

init(1M)

NAME

`init, telinit -` process control initialization

SYNOPSIS

`/sbin/init [ 0123456SsQqabc ]`

`/sbin/telinit [ 0123456SsQqabc ]`

DESCRIPTION

init

`init` is a general process spawner. Its primary role is to create processes from information stored in the file `/etc/inittab` [see `inittab(4)`].

At any given time, the system is in one of eight possible run levels. A run level is a software configuration of the system under which only a selected group of processes exist. The processes spawned by `init` for each of these run levels is defined in `/etc/inittab`. `init` can be in one of eight run levels, `0-6` and `S` or `s` (run levels `S` and `s` are identical). The run level changes when a privileged user runs `/sbin/init`. This user-spawned `init` sends appropriate signals to the original `init` spawned by the operating system when the system was booted, telling it the run level to which it should change.

The following are the arguments to `init`.

`0` Shut the machine down so it is safe to remove the power.

`1` Put the system in system administrator mode. All file systems are mounted. Only a small set of essential kernel processes are left running. This mode is for administrative tasks such as installing optional utility packages. All files are accessible and no users are logged in on the system.

`2` Put the system in multiuser mode. All multiuser environment terminal processes and daemons are spawned. This state is commonly referred to as the multiuser state.

`3` Start the remote file sharing processes and daemons. Mount and advertise remote resources. Run level `3` extends multiuser mode and is known as the remote-file-sharing state.

`4` Is available to be defined as an alternative multiuser environment configuration. It is not necessary for system operation and is usually not used.

`5` Stop the Reliant UNIX system and switch to the SASH (stand-alone shell) [for RM600] or in the PROM monitor [for RM400].

`6` Stop the Reliant UNIX system and reboot to the state defined by the `initdefault` entry in `/etc/inittab`.

The system boot can be interrupted on the RM600 in accordance with the boot options [cf. `bootflags(8)`].

`a, b, c` Process only those `/etc/inittab` entries having the `a`, `b`, or `c` run level set. These are pseudo-states, which may be defined to run certain commands, but which do not cause the current run level to change.

`Q, q` Re-examine `/etc/inittab`.

`S, s` Enter singleuser mode. When this occurs, the terminal which executed this command becomes the system console. This is the only run level that doesn't require the existence of a properly formatted `/etc/inittab` file. If this file does not exist, then by default the only legal run level that `init` can enter is the singleuser mode. If this file exists, init looks for commands in `/etc/inittab` pertaining to singleuser state. When the system comes up to `S` or `s`, file systems for users' files are not mounted and only essential kernel processes are running. When the system comes down to `S` or `s`, all mounted file systems remain mounted, and all processes started by `init` that should only be running in multiuser

mode are killed. In addition, any process that has a `utmp` entry will be killed. This last condition insures that all port monitors started by the SAC are killed and all services started by these port monitors, including `ttymon` login services, are killed. Other processes not started directly by `init` will remain running. For example, `cron` remains running.

When a Reliant UNIX system is booted, `init` is invoked and the following occurs. First, `init` looks in `/etc/inittab` for the `initdefault` entry [see inittab(4)]. If there is one, `init` will usually use the run level specified in that entry as the initial run level to enter. If there is no `initdefault` entry in `/etc/inittab`, `init` requests that the user enter a run level from the virtual system console. If an `S` or `s` is entered, `init` goes to the singleuser state. In the singleuser state the virtual console terminal is assigned to the user's terminal and is opened for reading and writing. The command `/sbin/su` is invoked and a message is generated on the physical console saying where the virtual console has been relocated. Use either `init` or `telinit`, to signal `init` to change the run level of the system. Note that if the shell is terminated (via an end-of-file), `init` will only re-initialize to the singleuser state if the `/etc/inittab` file does not exist.

If a `0` through `6` is entered, `init` enters the corresponding run level. Run levels `0`, `5`, and `6` are reserved states for shutting the system down. Run levels `2`, `3`, and `4` are available as multiuser operating states.

If this is the first time since power up that `init` has entered a run level other than singleuser state, `init` first scans `/etc/inittab` for `boot` and `bootwait` entries [see inittab(4)]. These entries are performed before any other processing of `/etc/inittab` takes place, providing that the run level entered matches that of the entry. In this way any special initialization of the operating system, such as mounting file systems, can take place before users are allowed onto the system. `init` then scans `/etc/inittab` and executes all other entries that are to be processed for that run level.

To spawn each process in `/etc/inittab`, `init` reads each entry and for each entry that should be respawned, it forks a child process. After it has spawned all of the processes specified by `/etc/inittab`, `init` waits for one of its descendant processes to die, a powerfail signal, or a signal from another `init` or `telinit` process to change the system's run level. When one of these conditions occurs, `init` re-examines `/etc/inittab`. New entries can be added to `/etc/inittab` at any time; however, `init` still waits for one of the above three conditions to occur before re-examining `/etc/inittab`. To get around this, `init Q` or `init q` command wakes `init` to re-examine `/etc/inittab` immediately.

When `init` comes up at boot time and whenever the system changes from the singleuser state to another run state, `init` sets the ioctl(2) states of the virtual console to those modes saved in the file `/etc/ioctl.syscon`. This file is written by `init` whenever the singleuser state is entered.

When a run level change request is made `init` sends the warning signal (`SIGTERM`) to all processes that are undefined in the target run level. `init` waits five seconds before forcibly terminating these processes via the kill signal (`SIGKILL`).

When `init` receives a signal telling it that a process it spawned has died, it records the fact and the reason it died in `/var/adm/utmp` and `/var/adm/wtmp` if it exists [see who(1)]. A history of the processes spawned is kept in `/var/adm/wtmp`. If `/var` is not accessible, then `init` records this information in temporary files in `/stand` and saves this information when `/var` becomes accessible (no information is lost).

If `init` receives a `powerfail` signal (`SIGPWR`) it scans `/etc/inittab` for special entries of the type `powerfail` and `powerwait`. These entries are invoked (if the run levels permit) before any further processing takes place. In this way `init` can perform various cleanup and recording functions during the powerdown of the operating system.

telinit

`telinit`, which is linked to `/sbin/init`, is used to direct the actions of `init`. It takes a one-character argument and signals `init` to take the appropriate action.

DIAGNOSTICS

If `init` finds that it is respawning an entry from `/etc/inittab` more than ten times in two minutes, it will assume that there is an error in the command string in the entry, and generate an error message on the system console. It will then refuse to respawn this entry until either five minutes has elapsed or it receives a signal from a user-spawned `init` or `telinit`. This prevents `init` from eating up system resources when

someone makes a typographical error in the `inittab` file or a program is removed that is referenced in `/etc/inittab`.

When attempting to boot the system, failure of `init` to prompt for a new run level may be because the virtual system console is linked to a device other than the physical system console.

NOTES

`init` and `telinit` can be run only by a privileged user.

`init` returns from singleuser state without `/var` being mounted. Typically, `init` executes a script of the form `/sbin/rc`*N* to go into state *N* (via `/etc/inittab`). These scripts execute scripts of the form `/etc/rc`*N*`.d/[KS]*`. `/sbin/rc`*N* scripts have been modified to recognize an error status of `100` from the `/etc/rc`*N*`.d/[KS]*` scripts as a major failure and cause the system to go into singleuser mode.

The `S` or `s` state must not be used indiscriminately in the `/etc/inittab` file. A good rule to follow when modifying this file is to avoid adding this state to any line other than the `initdefault`. However, this varies from site to site. More items may be done in singleuser state by updating `/etc/inittab` with proper entries, for example `/sbin/rcS` for site-specific singleuser state script.

If a default state is not specified in the `initdefault` entry in `/etc/inittab`, state `0` is entered. The system then switches off.

If the `utmp` file cannot be created in the `/var` filesystem when booting the system, a `utmp` file is created in the `/stand` file system. This is later moved to the `/var` file system when it is mounted. This happens transparently to the system administrator. The same result is achieved whether the `/var` filesystem is mounted or not when booting the system.

If the system allows kernel to pass boot-time arguments to `init` (main `init`), the following arguments can be passed.

`-l #` The `-l` option takes a number which denotes the `init` run-level. It accepts only: `123456sS`, and defaults to singleuser if any other value is specified.

`-s, -S`
Go to singleuser level.

`-a` Ask for new level, ignoring level defined by the `initdefault` entry in `/etc/inittab`.

`-n` A new shell is created before going into any level. If used with the singleuser level, this happens much before singleuser mode is entered and can be used to repair things by the system administrator.

`-x #` This avoids the debug level, causing `init` to append entries in `/etc/debug`. This file is always appended and not removed by `init`. The file should be removed once `init` is debugged, or after modifying entries in `/etc/inittab`.

`-v #` This sets `INIT_VERBOSE`=1 in global environment for children of `init`.

FILES

`/etc/inittab`
`/var/adm/utmp`
`/var/adm/wtmp`
`/etc/ioctl.syscon`
`/dev/console`

SEE ALSO

`login(1)`, `sh(1)`, `stty(1)`, `who(1)`, `ttymon(1M)`, `shutdown(1M)`, `kill(2)`, `inittab(4)`, `utmp(4)`, `utmpx(4)`, `termio(7)`.

System Administrator's Reference Manual

initdumpfs(1M)

NAME

`initdumpfs -` create file system for large memory dumps

SYNOPSIS

`initdumpfs [ NONE | [–f ] partition]`

DESCRIPTION

Memory dumps can be very large on systems that have an EHIOS board installed. These memory dumps should be accommodated in a single file system. The `initdumpfs` command supports the system administrator in creating this file system and performs consistency checks.

When called without options, `initdumpfs` outputs the current settings in relation to the memory dump and interactively prompts the system administrator for the partition where the file system is to be created.

`initdumpfs` performs the following checks:

- Are there any memory dumps under `/var/crash`?
- Is the partition being used for any other purposes?
- Is the data area behind the partition at least 250 Mbytes in size or, in the case of large memory configurations, is it greater than one sixth the size of the main memory?

If no errors are established, a `mkfs` command is executed for this partition and the partition is mounted as a file system under `/var/crash`. The relevant entries for automatic actions following the reboot are made in the respective configuration files.

If you specify the argument `NONE` for `initdumpfs`, the system administrator decides that no further memory dumps are to be saved. This means that panics can no longer be analyzed.

If the `–f` (forced) option is used or if the *partition* is specified directly, `initdumpfs` does not run interactively.

`initdumpfs` initializes several parameters from `/etc/default/dumpsave`:

`DUMPDIR=/var/crash`
`DUMPARTITION=`*partition* or `NONE`

DIAGNOSTICS

The `initdumpfs` command has the return code zero, if the settings were made correctly; otherwise, a return code other than zero is supplied and an appropriate error message is issued.

Error codes:

`1` Interactive mode aborted.
`2` Error creating file system.
`3` Files available under `/var/crash`.
`4` Incorrect number of parameters.
`5` Bad partition specification.
`6` No suitable device entry for the partition.
`8` Data area for the partition too small.

FILES

`/etc/default/dumpsave`

SEE ALSO

`dumpsave(1M)`, `mkfs(1M)`, `dumpsave(4)`.

System Administrator's Reference Manual

initlog(1M)

NAME

`initlog` - log console output

SYNOPSIS

`/sbin/initlog` *shellscript*

DESCRIPTION

The `initlog` program logs the output from any shell script. It was created specially to log all console output at startup and shutdown. The output of the specified shell script is duplicated using pseudo terminals and redirected to `/dev/osm` if necessary [see `osm`(7)].

`initlog` is terminated once the shell script has run, unless the shell script starts daemon processes in the background without redirecting the output. In this case, the standard output process is only terminated by `initlog` if no other process has opened the standard output channel.

The output is given in lines to the `/dev/osm` driver for logging. Every log entry receives a time stamp by default and a signature (name of shell script).

ARGUMENTS

shellscript
: Any executable shell script created by the user which can be assigned options and arguments.

The /etc/default/initlog file

The parameterization of `initlog` is controlled by means of the `/etc/default/initlog` file. The parameters in this file are structured as follows:

Format 1:

`VARIABLE`=*value*

Format 2:

shellscript`_VARIABLE`=*value*

Format 1 settings are global while Format 2 settings, on the other hand, are only effective for the specified shell script but have a higher priority than global settings.

Possible `VARIABLE` entries include:

`SHELL`
: Specifies the required shell. If a shell is not specified, `/sbin/sh` is entered as the default.

`TIMESTAMP`
: A time stamp is output before each logged line. Possible values include:

  `0` no time stamp

  `1` with time stamp (default)

`LABEL`
: Specifies a signature in the log line. The name of the shell script is taken as the default. To suppress the signature, no entries are made following the keyword.

`LOGFILE`
: Specifies an alternative log file. The `/dev/osm` driver is the default.

The log is turned off as soon as a control character (ESC sequence) is output by the script (e.□g. by means of `vi` or `sysadm`).

Logging can also be turned on and off explicitly. Therefore, shell script has to output the following key terms including line wrap (\n):

`InItLoG OfF`
: Logging off

`InItLoG On`
: Logging on

The transferred parameters are forwarded to the shell script.

The log entries in `/dev/osm` are taken over by system logging.

EXAMPLES

The following variable is set in the `/etc/default/initlog` file:

```
tsh_SHELL=/sbin/ksh
```

The user's `tsh` script contains the following information:

```
#!/sbin/sh
echo Hello World
```

When `tsh` has been executed, the following message is output on the screen:

```
Hello World
```

There is no logging in this case.

However, if the user's `tsh` script contains the following first line:

```
#!/sbin/initlog
echo Hello World
```

the following message is likewise output on the screen following execution of `tsh`:

```
Hello World
```

But logging also takes place in the `/var/adm/log/osm` file [see `osm.conf(4)`]. The

```
tail –1 /var/adm/log/osm
```

command then reports, for example:

```
Jun 27 11:43:48 tsh Hello World
```

The log line is preceded by the time stamp (default) and the signature (default: name of the shell script).

NOTES

The `initlog` command is used for logging the output during system initialization.

The `/etc/default/initlog` file is optional and by default does not exist.

The output of the shell script executed with `initlog` and its subscripts or programs should only go to standard input, standard output and standard error (`stdin`, `stdout`, `stderr`).

WARNING

Different input methods should not be used for the same input device (e.g. `/dev/console` and `stdin`). In this case, the input stream would be distributed arbitrarily to both channels.

FILES

```
/etc/default/initlog
/dev/osm
/var/adm/log/osm
/sbin/sh
```

SEE ALSO

`ksh(1)`, `sh(1)`, `syslogd(1M)`, `osm.conf(4)`, `console(7)`, `osm(7)`.

System Administrator's Reference Manual

installf(1M)

NAME

`installf -` add a file to the software installation database

SYNOPSIS

`installf` [`–c` *class*] *pkginst pathname* [*ftype* [[*major minor*] [*mode owner group*]]

`installf` [`–c` *class*] *pkginst* `–`

`installf –f` [`–c` *class*] *pkginst*

DESCRIPTION

`installf` informs the system that a pathname not listed in the `pkgmap` file is being created or modified. It should be invoked before any file modifications have occurred.

When the second synopsis is used, the pathname descriptions will be read from standard input. These descriptions are the same as would be given in the first synopsis but the information is given in the form of a list. The descriptions should be in the form:

pathname [*ftype* [[*major minor*] \c [*mode owner group*]].

After all files have been appropriately created and/or modified, `installf` should be invoked with the `–f` synopsis to indicate that installation is final. Links will be created at this time and, if attribute information for a pathname was not specified during the original invocation of `installf` or was not already stored on the system, the current attribute values for the pathname will be stored. Otherwise, `installf` verifies that attribute values match those given on the command line, making corrections as necessary. In all cases, the current content information is calculated and stored appropriately.

OPTIONS

`–c` *class*
: Class to which installed objects should be associated. Default class is `none`.

pkginst
: Name of package instance with which the pathname should be associated.

pathname
: Pathname that is being created or modified.

ftype A one-character field that indicates the file type. Possible file types include:

- `f` astandard executable or data file
- `e` afile to be edited upon installation or removal
- `v` volatile file (one whose contents are expected to change)
- `d` directory
- `x` an exclusive directory
- `l` linked file
- `p` named pipe
- `c` character special device
- `b` block special device
- `s` symbolic link

major The major device number. The field is only specified for block or character special devices.

minor The minor device number. The field is only specified for block or character special devices.

mode The octal mode of the file (for example, 0664). A question mark (`?`) indicates that the mode will be left unchanged, implying that the file already exists on the target machine. This field is not used for linked or symbolically linked files.

owner The owner of the file (for example, `bin` or `root`). The field is limited to 14 characters in length. A question mark (`?`) indicates that the owner will be left unchanged, implying that the file already exists on the target machine. This field is not used for linked or symbolically linked files.

group The group to which the file belongs (for example, `bin` or `sys`). The field is limited to 14 characters in length. A question mark (`?`) indicates that the group will be left unchanged, implying that the file already exists on the target machine. This field is not used for linked or symbolically linked files.

`–f` Indicates that installation is complete. This option is used with the final invocation of `installf` (for all files of a given class).

NOTES

When *ftype* is specified, all applicable fields, as shown below, must be defined:

ftype	Required Fields
`p x d f v` or `e`	*mode owner group*
`c` or `b`	*major minor mode owner group*

`installf` without `–f` will create directories, named pipes and special devices on the original invocation. Links are created when `installf` is invoked with the `–f` option to indicate installation is complete.

Links should be specified as *path1=path2* instead of *pathname*. *path1* indicates the destination and *path2* indicates the source file.

Files installed with `installf` will be placed in the class *none*, unless a class is defined with the command. Subsequently, they will be removed when the associated package is deleted. If this file should not be deleted at the same time as the package, be certain to assign it to a class which is ignored at removal time. If special action is required for the file before removal, a class must be defined with the command and an appropriate class action script delivered with the package.

When classes are used, `installf` must be used as follows:

```
installf -c class1 ...□
installf -f -c class1 ...□
installf -c class2 ...□
installf -f -c class2 ...
```

EXAMPLES

The following example shows the use of `installf` invoked from an optional `preinstall` or `postinstall` script:

```
#create /dev/xt directory□
#(needs to be done before drvinstall)□
installf $PKGINST /dev/xt d 755 root sys || exit 2□
majno=`/usr/sbin/drvinstall –m /etc/master.d/xt□
      –d $BASEDIR/data/xt.o –v1.0` || exit 2□
i=00□
while [ $i –lt $limit ]□
do□
      for j in 0 1 2 3 4 5 6 7□
      do□
             echo /dev/xt$i$j c $majno `expr $i * 8 + $j`□
                       644 root sys |□
             echo /dev/xt$i$j=/dev/xt/$i$j□

   done□
      i=`expr $i + 1`□
      [ $i –le 9 ] && i="0$i" #add leading zero□
```

```
done | installf $PKGINST - || exit 2
# finalized installation, create links
installf –f $PKGINST || exit 2
```

SEE ALSO

pkgadd(1M), pkgask(1M), pkgchk(1M), pkginfo(1M), pkgmk(1M), pkgparam(1M), pkgproto(1M), pkgtrans(1M), pkgrm(1M), removef(1M).

System Administrator's Reference Manual

iotail(1M)

NAME

`iotail` - IO request statistics monitor

SYNOPSIS

`iotail` [`-ynptvADO`] [`-m` *num*] [*device*]

DESCRIPTION

`iotail` continuously prints the service times (in micro-seconds) of IO requests as they are completed for a single device. Typically, this information is used to produce graphs describing the performance characteristics of an IO subsystem or to follow the device usage pattern of an application.

Any number of `iotail` commands can be simultaneously active on a system for any combination of traceable devices. This includes several invocations of `iotail` tracing the same device.

By default, `iotail` will display requests as:

ssssssss + *dddddddd* = *ttttttt C* □*lllllll OOOOOOOO*

Where:

ssssssss
: is the length of time the request remained queued in the device driver before it was passed on to the device.

dddddddd
: is the length of time the device took to complete the request.

ttttttt
: is the sum of *ssssssss* and *dddddddd*.

C is `R` if it was a read or receive request and is `W` if it was a write or send request.

lllllll
: is the length of the request in bytes.

OOOOOOOO
: is device-specific information that may be present for some types of devices. For disks the format of this information is as follows:

[*cccc*:*hhhh*:*ssss*]

Where:

cccc is the cylinder number.

hhhh is the head or track number.

ssss is the sector.

OPTIONS

`-y` turns on request tracing for *device*.

`-n` turns off request tracing for *device*.

`-p` continuously displays traced requests for *device*.

`-t` prints a table of traceable devices.

`-v` is verbose when printing traceable devices using `-t`.

`-m` *num*
: For efficiency, `iotail` waits until *num* requests have occurred before displaying them. The default is 5 requests.

`-A` prints absolute time values (in micro-seconds) relative to an arbitrary zero point. The times displayed

are:

`start`
when the kernel device driver receives the request.

`devstart`
when the request is sent to the device.

`done` when the request completes or the system receives the completion interrupt.

`-D` prints on each line the difference between the completion time of the request on the previous line and the device start time of the request on the current line.

`-O` Displays the device specific opaque data numerically.

SEE ALSO

`dkstat`(1M) .

keyload(1M) - lpusers(1M)

System Administrator's Reference Manual

keyload(1M)

NAME

`keyload -` load keyboard tables

SYNOPSIS

`/sbin/keyload` [`-d` *terminal*] [`-t` *type*] [`-l` *label*] [`-c` *comptable*] [*table*]

DESCRIPTION

The `keyload` command sets the keyboard of a 97801 or 97808 terminal for different national languages by loading key assignments from corresponding keyboard tables.

The terminal's line discipline does not have to be set by the user, as it is appropriately modified and restored by `keyload`. If `keyload` is called without its *table* argument, it tries to find information in the `/etc/termtab` file (LK string).If it does not succeed, it reports an error.

It is also possible to load compose and dead key tables (`-c` option). If you are using an 8-bit terminal you must make sure that the line to the terminal has 8-bit capability (if you are working with `rlogin`, for example).

OPTIONS

`-d` *terminal*
terminal is the path name of the special file used to reach the terminal, e.g. `/dev/term/tty003`. The path name can also be given relative to the `/dev` directory, such as `term/tty003`.

`-t` *type*
With this option you specify the type of terminal associated with the keyboard. If this option is not specified, `keyload` does not check that the table type is suitable for the terminal. *type* can be any of the following:

`new` 7-bit terminals

`7new` 8-bit terminals running in 7-bit mode

`8bit` 8-bit terminals running in 8-bit mode

`-l` *label*
To establish whether an 8-bit terminal is running in 7-bit or 8-bit mode, you can specify a label from the `/etc/gettydefs` file (e.g. S, N). Then on the basis of the entry associated with the given label `keyload` determines whether the terminal is configured. If `-l`*label* is specified `keyload` does not check how the terminal is actually set.

`-c` *comptable*

This option lets you specify the (path) name of a compose or dead key table. If *comptable* is not a full path name, `keyload` expects to find the files to load in the `/etc/keytables` directory, ending with the suffixes `.comp` and `.dead`. Thus `-c greek` tells `keyload` to look for `/etc/keytables/greek.comp` and `/etc/keytables/greek.dead`. If you want the tables to be taken from files in some other directory, you have to give the full path name. The escape sequences that the terminal has to be sent before and after the compose and dead key tables proper are inserted by keyload and must not be included in the files.

table

Name of the keyboard table to be loaded, e.g. `span` for Spanish or `ital` for Italian. The possible values can be determined from the file names in the `/etc/keytables` directory.

Matching keyload to the terminal

`keyload` tries to identify old terminals on the basis of their firmware version string (on new terminals there is a special command for checking terminal capabilities). There are a few cases of terminals with special version strings not contained in `keyload`'s internal table. `keyload` can be matched to such terminals by setting up a configuration file called `/etc/keytables/key_fw`.

The terminal capability information is in two parts: the firmware version string in the first column and one of the digits 0, 7 or 8 in the second, where 0 means that the terminal does not have a downloadable keyboard and 7 and 8 mean that the terminal is respectively 7-bit or 8-bit and has a downloadable keyboard. Any other character in the second column is ignored. In the version string in the first column the letter `x` at any position means that that position is to be ignored. The two columns must be separated by at least one blank or tab.

Examples:

The following entry means that terminals reporting a version string of `830500` are 8-bit terminals:

```
830500 8
```

The following entry tells `keyload` that terminals with version strings beginning with `55` do not have downloadable keyboards.

```
55xxxx 0
```

You can query the firmware version string of a terminal by sending it the escape sequence `ESC[4y`.

Compatibility with earlier versions of the keyload program

`keyload` can also be invoked with its old parameters:

`keyload` [`-t` *type*] [*terminal*] *table*

If the `-c` option is not used, `keyload` looks for compose and dead key tables in the `/etc/ckeytables` and `/etc/dkeytables` directories under the same name as *table*. With this method the tables must include the escape sequences required before and after the tables themselves. This approach is not as flexible as using the `-c` option.

EXAMPLES

The following command downloads the French keyboard table to the keyboard of the terminal at which the command is entered. `keyload` itself works out which special file to use and which type of terminal is involved:

```
$ keyload franz
```

`keyload` sends the `/etc/keytables/inter.new` table to the `/dev/term/tty003` terminal without further checking the terminal type:

```
$ keyload -t new -d term/tty003 inter
```

FILES

`/etc/ckeytables`
`/etc/dkeytables`

Directories searched for compose and dead key tables if the `-c` option is not used.

`/etc/keytables/key_fw`

File in which you can store a list of assignments associating terminal firmware versions with terminal capabilities.

`/etc/keytables`
Directory containing the keyboard tables for the different native languages.

SEE ALSO

`gettydefs`(4), `getty`(8).

System Administrator's Reference Manual

killall(1M)

NAME

`killall` - kill all active processes

SYNOPSIS

`/usr/sbin/killall` [*signal*]

DESCRIPTION

`killall` is used by `/usr/sbin/shutdown` to kill all active processes not directly related to the shutdown procedure.

`killall` terminates all processes with open files so that the mounted file systems will be unbusied and can be unmounted.

`killall` sends *signal* [see `kill(1)`] to all processes not belonging to the above group of exclusions. If no *signal* is specified, a default of `15` (`SIGTERM`) is used.

The `killall` command can be run only by the superuser.

LOCALE

The `LC_MESSAGES` environment variable governs the language in which message texts are displayed.

If `LC_MESSAGES` is undefined or is defined as the null string, it defaults to the value of `LANG`. If `LANG` is likewise undefined or null, the system acts as if it were not internationalized.

If any of the locale variables has an invalid value, the system acts as if none of the variables was set.

The `LC_ALL` environment variable governs the entire locale. `LC_ALL` takes precedence over all the other environment variables that affect internationalization.

FILES

`/usr/sbin/shutdown`

SEE ALSO

`kill(1)`, `ps(1)`, `fuser(1M)`, `shutdown(1M)`, `signal(2)`.

System Administrator's Reference Manual

labelit(1M)

NAME

`labelit(generic)` - provide labels for file systems

SYNOPSIS

`labelit` [`–F` *FSType*] [`–V`] [`–o`*specific_options*] *special* [*operands*]

DESCRIPTION

`labelit` can be used to provide labels for unmounted disk file systems or file systems being copied to tape.

The *special* name should be the physical disk section (e.g. `/dev/ios0/rsdisk000s3`), or the tape (e.g. `/dev/ios0/rstape00`). The device may not be on a remote machine. *operands* are *FSType*-specific and the manual page of the *FSType*-specific `labelit` command should be consulted for a detailed description.

There are two types of option: *generic_options* and *specific_options*. *generic_options* indicate options that are supported not only by the generic but also by the *FST*-specific module of the command (if that command exists). *specific_options* indicate suboptions that are supported only by the *FST*-specific module of the command. They are specified in a comma-separated list and/or as keyword-attribute pairs for interpretation by the *FST*-specific module of the command. These suboptions are specified via the `–o` option.

OPTIONS

The *generic_options* are:

`–F` Specifies the *FSType* on which to operate. The *FSType* should either be specified here or be determinable from `/etc/vfstab` by matching the *special* with an entry in the table.

`–V` Echoes complete command line. This option is used to verify and validate the command line. Additional information obtained via a `/etc/vfstab` lookup is included in the output. The command is not executed. When used with the *FSType* option, it passes the `–V` to the *FSType*-specific module for interpretation.

`–o` Specifies *FSType*-specific options.

There are no *other_options* for this command.

NOTE

This command may not be supported for all FSTypes.

FILES

`/etc/vfstab`
List of default parameters for each file system

SEE ALSO

`vfstab`(4) .

Manual pages for the FSType-specific modules of `labelit`.

System Administrator's Reference Manual

labelit(1M-ufs)

NAME

`labelit(ufs)` - provide labels for `ufs` file systems

SYNOPSIS

`labelit` [`-F ufs`] [*generic_options*] *special* [*fsname volume*]

DESCRIPTION

`labelit` can be used to provide labels for unmounted `ufs` disk file systems or `ufs` file systems being copied to tape.

With the optional arguments omitted, `labelit` prints current label values.

The *special* name should be the physical disk section (e.g. `/dev/ios0/rsdisk000s3`), or the tape (e.g. `/dev/ios0/rstape00`). The device may not be on a remote machine.

The *fsname* argument represents the mounted name (e.g. `root`, `u1`, etc.) of the file system.

volume may be used to equate an internal name to a volume name applied externally to the disk pack, or tape.

For file systems on disk, *fsname* and *volume* are recorded in the superblock.

OPTIONS

`-F ufs`
: Specifies the `ufs` FSType. Used to ensure that an `ufs` file system is labeled.

generic_options
: Options supported by the generic `labelit` command.

SEE ALSO

generic `labelit`(1M), `mount`(1M-ufs), `fs`(4).

System Administrator's Reference Manual

labelit(1M-vxfs)
VxFS

NAME

`labelit(vxfs)` - provide labels for a Veritas File System (VxFS)

SYNOPSIS

`labelit` [`–F vxfs`] [*generic_options*] [`–n`] *special* [*fsname volume*]

DESCRIPTION

This manual page is an addendum to the `labelit(1M)` manual page. It describes the Veritas File System (VxFS) specific features of the `labelit` command.

`labelit` can be used to provide labels for unmounted Veritas disk file systems or for tapes being used to copy Veritas file systems. With the optional arguments omitted, `labelit` prints current label values.

The *special* name should be the disk partition (for example, `/dev/ios0/rsdisk000s3`) or the tape device (for example, `/dev/ios0/rstape000`). The device may not be on a remote machine.

The *fsname* argument represents the mounted name (for example, `/`, `/usr`, and so on) of the file system. *volume* may be used to equate an internal name to a volume name applied externally to the disk pack, diskette or tape.

For file systems on disk, *fsname*, and *volume* are recorded in the super block.

OPTIONS

`–F vxfs`
: Specifies the VxFS file system type (*FSType*). Used to ensure that a VxFS file system is labeled.

generic_options
: Options supported by the generic `labelit` command.

`–n`
: For initial tape labeling only [to prepare tapes for `volcopy(1M)`]. This destroys the previous contents of the tape.

SEE ALSO

`labelit(1M)`, `volcopy(1M)`, `volcopy(1M-vxfs)`, `fs(4-vxfs)`.

System Administrator's Reference Manual

launchit(1M)

NAME

`launchit` - launch a process with special capabilities

SYNOPSIS

`launchit` [–g*name*] [–u*name*] [–d*n*] [–c*m*] [–C*m*] [–p*x*] [–P*x*] [–n{e|d}] [–D] [–U] [–A] [–v] *commandstring* | *pid*

DESCRIPTION

WARNING:

The use of some of the options offered by this command may severely disrupt the normal functioning of the system.

`launchit` sets the special attributes specified by the option list and launches *commandstring* via an `execvp(2)` .

OPTIONS

–g*name*
: Immediately before launching *commandstring*, the process's group name is set to *name* (*name* is an ASCII group name).

–u*name*
: Immediately before launching *commandstring*, the process's user id is set to *name* (*name* is an ASCII user name).

–d*n* The process's file descriptor limit is set to *n*. *n* must be in the range 63 to 1024.

–c*m* The process is bound to CPU *m*.

Note: This option may be given multiple times to bind the process to a set of CPUs.

–C*m* The process is bound exclusively to CPU *m*.

Note: This option may be given multiple times to exclusively bind the process to a set of CPUs.

–p*x* The process will run at a fixed priority with a "nice" value of *x* [see `nice(2)`].

–P*x* The process will run in the fixed priority class with a priority of *x*.

–n{e|d}
: Enable or disable the support of null pointer dereferences. If enabled, the kernel will satisfy read accesses to address 0 by mapping a page of zeros read-only at address 0. This is enabled by default. This attribute is inherited by the processes children.

–D The process has the ability to disable preemption using `mpcntl(MPCNTL_NOPRMPT, ...)`.

–U The process will run with no limit on the number of children that may be created.

–A This process will run with the Reliant UNIX operating system's processor affinity mechanisms disabled.

–v Verbose mode. Print the current values of attributes set by `launchit`.

pid Process id as returned from `ps`. If the first letter of *commandstring* | *pid* is a digit, the parameter id taken as a pid, otherwise as a command to be executed.

SEE ALSO

`mpcntl(1M)` , `pfdisable(1M)` , `pfenable(1M)` , `pfpriv(1M)` , `setrlimit(2)` , `mpcntl(3X)` , `nullptr(3X)` .

System Administrator's Reference Manual

link(1M)

NAME

`link, unlink` - link and unlink files and directories

SYNOPSIS

`/usr/sbin/link` *file1 file2*

`/usr/sbin/unlink` *file*

DESCRIPTION

The `link` command is used to create a file name that points to another file. Linked files and directories can be removed by the `unlink` command; however, it is strongly recommended that the `rm` and `rmdir` commands be used instead of the `unlink` command.

The only difference between `ln` and `link` and `unlink` is that the latter two do exactly what they are told to do, abandoning all error checking. This is because they directly invoke the `link` and `unlink` system calls.

LOCALE

The `LC_MESSAGES` environment variable governs the language in which message texts are displayed.

If `LC_MESSAGES` is undefined or is defined as the null string, it defaults to the value of `LANG`. If `LANG` is likewise undefined or null, the system acts as if it were not internationalized.

If any of the locale variables has an invalid value, the system acts as if none of the variables was set.

The `LC_ALL` environment variable governs the entire locale. `LC_ALL` takes precedence over all the other environment variables that affect internationalization.

NOTES

These commands can be run only by the superuser.

SEE ALSO

`ln(1)`, `rm(1)`, `link(2)`, `unlink(2)`.

System Administrator's Reference Manual

listdgrp(1M)

NAME

`listdgrp -` list members of a device group

SYNOPSIS

`listdgrp` *dgroup ...*

DESCRIPTION

`listdgrp` displays the members of the device groups specified by the *dgroup* list.

DIAGNOSTICS

This command will exit with one of the following values:

`0` = Successful completion of the task.

`1` = Command syntax incorrect, invalid option used, or internal error occurred.

`2` = Device group table could not be opened for reading.

`3` = Device group *dgroup* could not be found in the device group table.

EXAMPLE

To list the devices that belong to group `partitions`:

```
$ listdgrp partitions
root
swap
usr
```

FILES

`/etc/dgroup.tab`

SEE ALSO

`putdgrp`(1M) .

System Administrator's Reference Manual

log3adm(1M)

NAME

log3adm - command for converting the log3admin administration file for Logging V3.0

SYNOPSIS

log3adm

DESCRIPTION

The log3adm command can be used to convert the log3admin(4) administration file, which contains the logbook definitions, the component assignments, the definition of the default logbook and the definition of the ASCII protocol file for Logging V3.0, to the binary log3admin.obj form. It is only if log3admin(4) contains no syntax errors that the binary version log3adm.obj is generated. log3adm.obj contains C structures, which are read in by the log3svr log server and the log3display(1M) and log3eval(1M) evaluation commands.

OPTIONS

The command has neither options nor usage.

DIAGNOSTICS

The return value is 0, is no error is detected. Standard output is used in this case:

"*n* LOGFILE's and *m* LOGTYPE's found."

If the log3adm command discovers an error situation, this is reported to standard error output and the return value is not 0.

1 "cannot open to read"

The log3admin file cannot be opened.

2 "memory allocation"

There is no free memory area for creating data structures in the main memory.

3 "cannot read"

The log3admin file cannot be read.

4 "cannot write"

The log3admin.obj file cannot be written.

5 "instruction missing/not correct"

There is an instruction missing. Syntactic error.

6 "name of logfile missing/not correct"

The logbook name is missing. Syntactic error.

7 "digit missing/not correct"

A digit is missing. Syntactic error.

8 "too many parameters"

There are too many parameters per line. Syntactic error.

9 "logfile used but not defined as #LOGFILE"

The logbook name is used in the component assignment although the logbook itself has not even been defined as yet. Semantic error.

10 "logfile already defined"

A logbook cannot be defined twice.

11 "logtype already writes in this logfile"

A component cannot be assigned twice, in other words a component may only write once to the same logbook.

NOTES

The file content is only checked for syntactic accuracy but not for admissibility, plausibility or completeness.

FILES

`$LOGPATH/log3admin`
`$ADMPATH/log3admin`
`$LOGPATH/log3admin.obj`
`$ADMPATH/log3admin.obj`

SEE ALSO

`log3display`(1M), `log3eval`(1M), `log3str`(1M), `log3admin`(4), `log3`(5).

System Administrator's Reference Manual

log3display(1M)

NAME

`log3display -` evaluate log data

SYNOPSIS

`$LOGPATH/bin/log3display` [`–cevlusSwh`] `–oX` [*logfile ...*]

DESCRIPTION

The `log3display` command selects the log data from log files, formats it using structure IDs, and displays it on the screen.

OPTIONS

`–c` *compno*[,*compno2*][,*comp1-comp2*]
: This option is used to select components at the evaluation stage. A maximum of 64 entries can be made.

  Example: `–c 1000,34,500-100,2,2222-3333`

`–e` *errno*[,*errno2*][,*err1-err2*]
: This option is used to select component error numbers (T-Log 2.0: subtype) at the evaluation stage. A maximum of 64 entries can be made.

  Example: `–e 1,5,10-65`

`–v` *level*[,*level2*][,*lev1-lev2*]
: This option is used to select the severity of errors at the evaluation stage. A maximum of 64 entries can be made.

  Example: `–v 0-9`

`–l` *yy*[*mm*[*dd*[*HH*[*MM*[*SS*]]]]]
: Any log data recorded earlier than the specified date is ignored.

  Example: `–l 92`

  This entry is synonymous with `920101000000`.

`–u` *yymmddHHMMSS*
: Any log data recorded after the specified date is ignored.

  Example: `–u 901231235959`

  The following is interpreted: 31. Dec. 1990 23:59:59

`–s` *applic_text*
: The name of a certain application can be specified with this option. No other applications are affected.

`–S` *structid*
: This option can be used to specify a structure ID, which corresponds to the structure ID (`STRUCT` statement) in the description file `log3struct(4)`.

`–w` *width*
: Line length of screen output. The line is generally `76` characters long. Other values can be defined using this option.

  Example: `–w 132`

`–h` The command syntax is displayed.

`–oX` Formatting of screen output. The log data is output on the screen (standard output `stdout`) as separate lines with a number of fields, with the fields on each line separated by the octal "\030" (CAN). The sequence and format of the fields is as follows:

  Date of entry *YYMMDD*

Time of entry	*hhmmss*
Date of last retry	*YYMMDD*
Time of last retry	*hhmmss*
Number of retry	*Number*
Current number	*Number*
User text=Module name	*Text*
Release version	*Text*
Severity of errors	*Number* or *text*
Error number	*Number*
Component number	*Number*
Process ID	*Number*
System name	*Text*
Message text	*Text*

All data is output as ASCII readable tests. All characters less 040 (blank) are mapped to blanks. Line feeds in the message text are flagged with "~".

logfile

This specification is used to select log files at the evaluation stage. If several entries are made, they must be separated by blanks (not commas).

If a log file is not specified, `log3display` transfers all log files defined under the `#LOGFILE` block in the administration file `log3admin(4)`.

If a specified file is not available, `log3display\f attaches all typical Logging V3.0 endings [0-8]log` one after another to the file names, and then tests these names.

FILES

`LOGPATH=/opt/log3`
Installation directory for Logging V3.0

`$LOGPATH/log3admin`
Configuration file: Assignment of applications to log files

`$LOGPATH/log3struct`
Configuration file: Assignment of structure IDs to output formats

`$LOGPATH/`*name*`[0-8]log`
Log files

SEE ALSO

`log3logger(1M)`, `syslogd(1M)`, `log3admin(4)`, `log3(5)`.

System Administrator's Reference Manual

log3do(1M)

NAME

log3do - DO command for Logging V3.0

SYNOPSIS

log3do –c *class-function* [–d *data*]

log3do –c logentryex –d *logentry* [–V *sep*] [–b]

log3do –c logstart | logrestart [–d *start-options*]

log3do –c logcountreset –d *unit-name*:*event-name*

log3do –c logeventsact [–d *list-of-unit-names*]

log3do –c logprotectevm [–d *unit-name*]

log3do –H

log3do –h

DESCRIPTION

The log3do command can be used to have various operations and functions executed within Logging V3.0. Some functions are closely linked to object classes while others have no unique affinity to an object class rather affect Logging V3.0 as a whole (e.g. restarting of Logging V3.0).

The log3do command is the DO command for Logging V3.0 that can be used to execute different operations or functions within Logging V3.0, which do not uniquely affect object classes, which affect several object classes at the same time or which are not one of the classic functions (create, modify and delete data records) within an object class.

Note:

Because the functions are not the same, the function calls use quite different call parameters. While these can be mapped to the same –d call option, they are interpreted differently for each function call. No plausibility, admissibility or validity check is performed for the data transferred in the call.

If no function class has been specified, the $CLASS environment variable is evaluated.

OPTIONS

–c *class-function*

This option determines the function that is to be executed. The following entries are permitted for *class-function*:

logstart
: Starts Logging

logstop
: Stops Logging

logrestart
: = logstop + logstart

logadminact
: Activates all administration data: logbookact + logcompact + logstructact

logbookact
: Activates logbook definitions only

logcompact
: Activates component assignments only

logstructact
: Activates structure definitions only

logcounterreset

Resets counters in Event Management

`logeventsact`
Activates event descriptions in Event Management

`logtvconv`
Converts event descriptions for TransView

`logentryex`
Fetches additional information for a log entry

`logprotectevm`
Lists description files with their protection

`logeventnames`
Lists event descriptions

`logreactionnames`
Lists reaction descriptions

`logweights`
Lists error weightings in Logging V3.0

`-d` *data*

The `-d` option is only permitted in conjunction with the following class functions for the `-c` option:

`-c logentryex -d` *logentry* [`-V` *sep*] [`-b`]
logentry is the complete log entry with attribute fields in default order as established by the `log3get(1M)` `GET` command: `@` is the default separator that connects all values.

If a different separator *sep* is chosen for transferring the log entry, this can be reconciled with the `-V` *sep* option. Only a single character is permitted. If the separator occurs in the data, it must be masked with "\".

The log entry is transferred as a call option to the functions that are to fetch the additional information. The `-b` option means that the only additional information established for the relevant log entry is that requested specifically by the user on the screen. The user is aware that it may take some time to get this additional information.

Example:

```
-d "943543494@943543494@0@@Logging Server@3.0A33@DEBUG@240@..."
```

`-c logstart` | `logrestart` [`-d` *start-options*]
The *start-options* are transferred unchanged as call options to the `rc2` start script `/etc/init.d/log3`. This involves a combination of letters either preceded fully (or each letter individually) by "-". Logging V3.0 is started or first stopped and then restarted.

Example: `-d "-aeTt -d0 -m 20"` or `-d "-a -e -T -t -d0 -m20"`

Valid start options include:

`-a` Send no log entries to Event Management.

`-e` Send no log entries to Event Reporting (DSM-ER).

`-T` Activate ASCII protocol.

`-t` Activate T-Log 2.0 support.

`-D` Activate debug mode for the log server and set the trace level to 1.

`-d` *level*
Set the trace level to 0 – 3.

`-m` *minutes*
Set the interval in minutes for logging time stamps.

`-c logcountreset -d` *unit-name*`:`*event-name*
The *unit-name* is the suffix for the associated event description file name: i.e. `cmn` for the `/opt/log3/filters/events.cmn` file.

The *event-name* is the respective text after the keywords "EVENT:□NAME=" from the event description file. The counter status for the specified unit is reset to 0.

Example:

```
–d "cmn:Memory Single Bit Error Limit Reached"
```

`–c logeventsact` [`–d` *list-of-unit-names*]

The changes in the `events.*` event description files are activated. The *unit-names* in the list must be separated with ":".

Example:

```
–d cmn:rdd:sadm:lce
```

`–c logprotectevm` [`–d` *unit-name*]

The protection (changeable, protocol, protected) for the specified unit is output.

Example:

```
–d cmn
```

`–H` The list of operations or function class names known to `log3do` is output (one name per line).

`–h` Usage for this command.

DIAGNOSTICS

The return value is 0 is no error is detected. If the `log3do` command discovers an error situation, this is reported to standard error output and the return value is not 0.

`1` "Illegal/less options!"

Bad call: Incorrect command options or option values were used.

`2` "Logging V3.0 not installed!"

The Logging V3.0 home directory cannot be established from the package information. This indicates that Logging V3.0 was not installed.

`3` "Logging V3.0 not correctly installed!"

Logging V3.0 has been installed but the necessary commands for reading out data (e.g. `log3eval`) cannot be found.

`4` "You are not authorised to administrate Logging V3.0!"

You are not authorized as a system administrator for Logging V3.0; protected data is therefore not available to you.

`5` "Unknown object class function "........."!"

The requested function does not exist or is not known to the current `log3do` call.

`7` "Illegal / no data ?!"

The necessary call data (`–d` option) was not transferred.

`10` "Logserver not found/not active!"

Logging V3.0 is not active or the `log3svr` log server cannot be found. Activations and functions that require the log server are therefore not possible (they may also not be necessary however!).

`11` "Error with log3adm!"

The content check for this administration file returned errors. The file must be corrected.

`12` "Error with log3struct and /etc/default/log3struct!"

The content check of at least one of the two administration files returned errors. The file(s) must be corrected.

`13` "Check events (".........") results in error!"

The checking of the named event and/or reaction description files returned errors. The file(s) must be corrected.

14 "Convert events (".....") results in error!"

The conversion of the named event and/or reaction description files returned errors. The file(s) must be corrected.

15 "Activate events results in error!"

The activation of the event and/or reaction description files returned errors.

NOTES

It is not reported as an error if requested data is not available. The output in this case is blank and the return value is 0.

FILES

LOGPATH=/opt/log3
$LOGPATH/log3admin
$LOGPATH/log3struct
$LOGPATH/default[0-8]log
$LOGPATH/*name*[0-8]log
$LOGBOOKS/*name*[0-8]log
$LOGPATH/log3status
$LOGPATH/filters/events.*
$LOGPATH/filters/reactions.*

SEE ALSO

log3eval(1M), log3get(1M), log3replace(1M), log3status(4), log3(5).

System Administrator's Reference Manual

log3eval(1M)

NAME

`log3eval -` logbook evaluation (unformatted)

SYNOPSIS

`log3eval` [`–M` *mode*] {`–c`|`–e`|`–l`|`–f`|`–t`|`–m`} [`–a` *attribute-name* | `–o` *attribute-char*] [`–w` *width*] [`–d` *sep*] [`–F` *format*] [`–z` *num*] [*logbook ...*]

`log3eval –h`

DESCRIPTION

The `log3eval` command can be used to have logbooks evaluated by Logging V3.0 [see `log3(5)`].

The `log3val` command replaces the `log3display(1M)` command (the latter continues to be supported) for evaluation of logbooks by Logging V3.0. While the `log3display` command was intended to act as a user interface (CLI = Command Line Interface) and to provide support for the FMLI interface, particular emphasis was placed on performance and flexible output with the `log3eval` command. In contrast to `log3display`, the attribute fields are unformatted, ASCII readable and output separately using a freely definable separator. The choice (also multiple output of the same columns) and the order of the columns can be selected freely. Use of this command is thus supported by the `GET` command for Logging V3.0 [see `log3get(1M)`]. The command is not intended for use as a CLI.

Note:

If the attribute columns requested are unknown, no error message is issued rather the columns remain blank. This means that the same set of attribute columns can be requested on systems running older releases of Logging V3.0 without error messages being issued.

OPTIONS

`–M` *mode*

This option determines the order of evaluation. The following entries are possible for *mode*:

`r` (reverse order) The most up-to-date file in the logbook is selected and evaluated in reverse order, in other words the most recent logbook entry is output first and the oldest last.

`d` (upward) The oldest file in the logbook cycle is selected and evaluated in forward order, in other words the oldest logbook entry is output first and the most recent entry last.

Note:

If a mode is specified explicitly, only one *logbook* logbook cycle can be specified (as otherwise the different cycles would have to be mixed). If no logbook cycle is specified, only the `default[0-8]log` default logbook is evaluated. If several logbook cycles are specified, a relevant error message is issued.

`–c`|`–e`|`–l`|`–f`|`–t`|`–m`

This option can be used to predefine certain selection criteria:

`–c` *component_number*
List of values and ranges separated by commas.

`–e` *error_number*
List of values and ranges separated by commas.

`–l` *error_weighting*
List of values and ranges separated by commas.

`–f` *YYMMDDhhmmss*
Start date/time (number of seconds since 1970).

`–t` *YYMMDDhhmmss*
End date/time (number of seconds since 1970).

–m *module_name*

Name (no regular expression).

Example:

```
–c 1080009,1080010-1080015 –e 1,5,100-110,1025 –f –m testcom
```

–a *attribute-name* | –o *attribute-char*

The –a or –o options can be used to define a selection of attribute columns and the order in which they will be output.

The information is provided in the form of a list of the attribute names [see log3get(1M)], separated by the @ character (a fixed separator, which does not occur in the attribute name).

A list of letters **without** separators can be specified for the attribute fields to be output instead of the list of attribute names.

The current attribute flags *attribute-char* and *attribute-name* are as follows:

Char	Name	Description
t	firstTime	Date/time of first occurrence of log entry
S	sequence	Serial number for all logbook entries
T	lastTime	Date/time of last occurrence of the log entry
R	repeatings	Number of repeats
O	objectID	Object entity (e.g. for printers of the same type, the name c printer)
m	moduleName	Module name
r	releaseVersion	Release and version of reporting component
l	errorWeight	Error weighting
e	errorNumber	Error number
c	componentNumber	Component number
C	componentName	Component name
p	processID	Process number (PID)
U	userID	User id
u	userName	User name
G	groupID	Group id
g	groupName	Group name
n	node_system	System name (if the message is from an external host, its n appended in brackets)
M	messageText	Actual log message text

Example of a list with attribute names:

```
–a firstTime@userName@messageText
```

Returns the time stamp, the user name and the log message text.

Example of a list with attribute IDs:

```
–o tuM
```

Returns the time stamp, the user name and the log message text.

Note:

If the –o and –a options are used at the same time, the last option is executed.

–w *width*
The line length can be defined for output in the range 40 to 2000 characters. The default is 2000 characters. Positive values outside this range are corrected to the limit values.

–d *sep*
Defines a specific separator *sep*. Only a single character is permitted. If this character occurs in the readable ASCII data, it is masked with "\". The default separator is CAN (octal 030), a non-printable character, which cannot occur in the readable ASCII data.

–F *format*
Defines an output format for the time stamp in the logbook entry. The following entries are permitted for *format*:

format	Output
0	*YYYYMMDDhhmmss*
1	*YYYYMMDD hhmmss*
2	*DD.MM.YYYY/HH:mm:ss*
3	*YYMMDDhhmmss*
4	*YYMMDD hhmmss*
5	*DD.MM.YY/hh:mm:ss*

If no output format is selected, the time in seconds since 1970 is output.

–z *num*
A maximum number of entries to be output can be defined. This is particularly interesting in combination with the mode entries `r` or `d` of the –M option, because a fixed number of the oldest or the most recent logbook entries is then output.

logbook ...
A list of logbooks can be specified. The names should be separated by blanks. Absolute path names can be specified or logbook names with and without a suffix "[0-8]log", which are then sought under $LOGPATH.

If no logbook names and mode are predefined, all logbooks under $LOGPATH are evaluated.

–h Usage for this command.

DIAGNOSTICS

The return value is 0 if no error is detected. If the log3eval command discovers an error situation, this is reported to standard error output and the return value is not 0. The messages are language-dependent.

1 "-<option> <value>: bad conversion!"

Bad call: Incorrect command options or option values were used.

2 Not used.

3 "Bad file log3struct.obj. Create a new one!"
"Bad file log3admin.obj. Create a new one!"

The administration files cannot be imported and must first be regenerated [see log3adm(1M) and log3str(1M)].

4 "Mode '%c' only in connection with a single logbook cycle!"

Bad call: The –M option can only be used in connection with a single logbook cycle.

If a logbook is not specified explicitly, only the default[0-8]log default logbook is evaluated.

5 "Logfile %s not found (attention: name without extension is requested)!"

The log file could not be found. This may be because a logbook name was specified with the *log suffix.

NOTES

The logbook comprises several logbook cycles that can be written to simultaneously. For evaluation, the log entries must be resorted in accordance with their time stamp and sequence number; this may take a considerable amount of time if the number of entries is sufficiently large. To avoid this, only the `default[0-8]log` default logbook, a `datasyslog[0-8]log` system logbook and the `log3debug[0-8]log` debug logbook are generally written and evaluated separately by Logging V3.0.

The file names are taken from the name of the cycle, which is defined in the `log3admin(4)` administration file, a cycle number and the "`log`" suffix. A cycle comprises at least two and at most nine files. The `log3admin` administration file defines the number of files that will make up each cycle. Accordingly, the cycle number varies between 0 and this defined maximum −1 (since counting begins at 0).

If no absolute path name is specified, the log files are sought under `$LOGPATH`. This is generally the home directory for Logging V3.0. If a different path is found under the `$LOGPATH` environment variables, it will be used. This means that the logbooks for the local system can be maintained anywhere in the system. By specifying absolute paths, logbooks can also be evaluated elsewhere (e.g. by other systems).

FILES

`LOGPATH=/opt/log3`

`$LOGPATH/log3admin`

`$LOGPATH/log3struct`

`$LOGPATH/default[0-8]log`

`$LOGPATH/`*`name`*`[0-8]log`

`$LOGBOOKS/`*`name`*`[0-8]log`

SEE ALSO

`log3display(1M)`, `log3get(1M)`, `log3(5)`.

System Administrator's Reference Manual

log3event_convert(1M)

NAME

`log3event_convert -` convert event description parameters to an object format

SYNOPSIS

`$LOGPATH/bin/log3event_convert` *Event-Description-File* ... [`–o` *Event-Object-File*]

DESCRIPTION

The `log3event_convert` command is used to convert event description parameters from the event description files to an event object file. The event description parameters written in plain text are converted to a format which is easy to read and to process for the event management daemon.

OPTIONS

Event-Description-File

The data in an event description file describes fully the event to be monitored. The output of the converted data is sent to the `$LOGPATH/filters/log3event.object` file. If the `LOGPATH` environment variable is not set, the default value `/opt/log3` is assumed.

`–o` *Event-Object-File*

The output of the converted data is sent to the specified event object file.

RETURN VALUES

The normal return value of the command is 0 and in the event of errors is not equal to 0. The syntactic and semantic format errors of the event description parameters in the event description files are sent to stderr.

EXAMPLES

```
log3event_convert events.sni events.ubft, ...
log3event_convert events.myhw -o myhw.object
```

FILES

```
$LOGPATH=/opt/log3
$LOGPATH/filters/log3event.object
$LOGPATH/bin/log3event_convert
$LOGPATH/bin/log3analyze
```

SEE ALSO

`log3get_pevman`(1M), `log3set_pevman`(1M), `log3`(5).

System Administrator's Reference Manual

log3get(1M)

NAME

`log3get - GET` command for Logging V3.0

SYNOPSIS

`log3get` [`–c` *object_class*] [`–a` *attribute_value...*] [`–s "`*selection*`"...`] [`–V` *sep*] [`–f` *num*] [`–t` *num*]

`log3get` [`–c` *object_class*] [`–H`]

`log3get –h`

DESCRIPTION

The `log3get` command can be used to have Logging V3.0 output all data. This includes both the log entries and all administration data, such as logbook definitions, event descriptions, runtime status etc.

The `log3get` command is the `GET` command for Logging V3.0, which can be used to fetch the unformatted data for all object classes. The information is output on a line by line basis for each data record. The attribute fields are separated by a freely definable separator. If the separator occurs in the data, it is masked with "\". The attributes can be numbered and ordered on the basis of a predefined list of attribute names. Specific data records can be selected by predefining selection criteria. You can output the relevant object classes using the `log3get –H` command. The attribute names can be output for each object class using the `log3get –c` *object_class* `–H` command.

Note:

If the attribute columns requested are unknown, no error message is issued rather the columns remain blank. This means that the same set of attribute columns can be requested on systems running any releases of Logging V3.0 without error messages being issued.

If no object class has been specified, the `$CLASS` environment variable is evaluated.

OPTIONS

`–c` *object_class*

This option determines the object class from which the data is to be read. The object class names known at present are as follows (see `–H` option):

`logentries`
: Logbook entries

`logprotocol`
: Definition of the protocol file for the ASCII protocol

`logdefault`
: Definition of the default logbook

`logbooks`
: Logbook definitions

`logcomponents`
: Component definition

`logstructures`
: Structure descriptions

`logevents`
: Event descriptions

`logreactions`
: Reaction descriptions

`logcounters`
: Counter states in Event Management

`logruntime`

Status of Logging V3.0

`logcycles`
Status of logbook cycles (number of entries, last access)

`-a` *attribute_value...*
The *attribute_value* argument can be used to define a selection of attribute columns and the order in which they will be output.

The information is provided in the form of a list of attribute names, separated by the `@` character (a fixed separator, which does not occur in the attribute name). The known attribute names can be output with the `log3get -c` *object_class* `-H` command.

Example: `log3get -c logentries -a firstTime@userName@messageText`

Returns the time stamp, the user name and the log message text of the log entries.

`-s "`*selection*`"...`
A list of expressions in the form *AttributenameOpValue* can be specified as a selection criterion for data records, where *Attributename* is one of the known attribute names (see `-H` option), *Op* is one of the operators ==, !=, ~=, <, <=, >, >= and *Value* is the requested attribute value. The selection criteria are linked in list form with "AND".

The operators can be surrounded by any number of blanks and tabs.

The entire list must be combined to form a single argument with "...".

Example:

```
-s "errorNumber == 1 AND messageTest ~= HINT"
```

`-V` *sep*
Defines a specific separator *sep*. Only a single character is permitted. If this character occurs in the readable ASCII data, it must be masked with "\". The default separator is CAN (octal 030), a non-printable character, which cannot occur in the readable ASCII data.

`-f` *num*
Predefines the line number where data output is to begin.

Example: `-f 201`

The 201st data record is output first.

`-t` *num*
Predefines the line number where data output is to end.

Example: `-t 100`

The first 100 data records at most are output.

`-H` If the option appears on its own, the list of object classes known to `log3get` is output (one name per line). If an object class name is specified in addition with the `-c` option, the list of attribute names for this object class is output (one line with the attribute names separated by the `@` character).

`-h` Usage for this command.

Note:

The `-f` and `-t` options allow the data records to be read out in sections. Caution is advisable if the number of data records changes dynamically and the line numbering is consequently not constant. Care is also needed with regard to the performance, which is not constant in relation to sections of a similar size. Considerably more data records have to be read out internally in order to output the required section.

DIAGNOSTICS

The return value is 0 if no error is detected. If the `log3get` command discovers an error situation, this is reported to standard error output and the return value is not 0.

`1` `"Illegal/less options!"`
Bad call: Incorrect command options or option values were used.

2 "Logging V3.0 not installed!"

The Logging V3.0 home directory cannot be established from the package information. This indicates that Logging V3.0 was not installed.

3 "Logging V3.0 not correctly installed!"

Logging V3.0 has been installed but the necessary commands for reading out data [e.g. log3eval(1M)] cannot be found.

4 "You are not authorised to administrate Logging V3.0!"

You are not authorized to perform administration of Logging V3.0; protected data is therefore not available to you.

5 "Unknown object class!"

The specified object class does not exist or is not known to the current log3get command.

NOTES

The speed at which the data can be established depends on the requested attribute columns and the specified selection criteria. If all of the known attribute columns are requested, no filtering generally takes place. In relation to the log entries, account is taken of the requested attribute columns and the way they are ordered as early as when the data is read out. Some specific selection criteria can likewise be considered for log entries when the data is read out. Additional filtering with awk is implied for all other object classes where specific attribute columns are requested and selection criteria are predefined.

The system name (node) is provided for all object classes as an attribute field (generally the second column), although it is really only included in the actual data records for the log entries.

FILES

LOGPATH=/opt/log3

$LOGPATH/log3admin

$LOGPATH/log3struct

$LOGPATH/default[0-8]log

$LOGBOOKS/*name*[0-8]log

$LOGPATH/*name*[0-8]log

$LOGPATH/protocol.[AB]

$LOGPATH/log3status

$LOGPATH/filters/events.*

$LOGPATH/filters/reactions.*

SEE ALSO

log3replace(1M), log3do(1M), log3eval(1M), log3status(4), log3(5).

System Administrator's Reference Manual

log3get_pevman(1M)

NAME

`log3get_pevman` - query threshold monitoring status in the event management daemon

SYNOPSIS

`$LOGPATH/bin/log3get_pevman –g all [ –v ]`

`$LOGPATH/bin/log3get_pevman –n` *Event-Name* `[ –v ]`

DESCRIPTION

The `log3get_pevman` command is used to query the current status of threshold monitoring in the event management daemon and to send the result to stderr. The event description parameters output are as follows:

`Event`, `Limit`, `Counter`, `Repeat-Counter`, `Error`, `Module`, `Component`, `Text`, `Timerelation`, `Object-Name`, `Object-Position`, `System`, `Reactions`

OPTIONS

`–n` *Event-Name*
: Only the event description parameters for the specified event are output.

`–g all`
: Event description parameters for all events currently being monitored by the event management daemon are output.

`–v`
: Two spaces are used as separators between the event description parameters. A minus sign is used to indicate that an event description parameter is missing.

  If the `–v` option is not specified, a CAN (= 0x18) character is used as a separator between the event description parameters or to indicate that the parameter is missing.

RETURN VALUES

The normal return value for the command is 0 in the event of errors and is not equal to 0.

EXAMPLES

```
log3get_pevman –v –n EVENT_DF01
log3get_pevman –g all
```

FILES

```
$LOGPATH=/opt/log3
$LOGPATH/bin/log3get_pevman
$LOGPATH/bin/log3analyze
```

SEE ALSO

`log3event_convert`(1M), `log3set_pevman`(1M), `log3`(5).

System Administrator's Reference Manual

log3logger(1M)

NAME

`log3logger -` send log information to the log server

SYNOPSIS

`$LOGPATH/bin/log3logger` [`-c` *comp_no*] [`-e` *err_no*] [`-l` *fault_level*] [`-s` *structid*] [`-t` *module_name*] [`-r` *applic_rel*] [`-h`] *loginfo* [*loginfo* ...]

DESCRIPTION

This command sends log information to the log server.

OPTIONS

`-c` *comp_no*
: Assigns the log message to the specified component number *comp_no*. Default: `0`.

`-e` *err_no*
: Assigns the specified error number *err_no* for the log message. Default: `0`.

`-l` *fault_level*
: Assigns the specified severity of errors *fault_level* to the log message. Default: `6` = `LOG_INFO`.

`-s` *structid*
: Formats the log message according to the specified structure ID *structid*. Default: `0` = `NO_STRUCT`.

`-t` *module_name*
: Supplies the log message with the specified module name *module_name*. Default: "`Logger`".

  Note: The module name may only comprise digits, letters and "_" (in particular it must not contain the ":" character).

`-r` *applic_rel*
: Supplies the log message with the specified release marker *applic_rel*. Default: "`3.0A00`".

`-h`
: Displays the `log3logger` syntax.

loginfo
: Log data record sent to the log server.

FILES

`LOGPATH=/opt/log3`
: Installation directory for Logging V3.0

`$LOGPATH/log3admin`
: Configuration file: Assignment of applications to log files

`$LOGPATH/log3struct`
: Configuration file: Assignment of structure IDs to output formats

`$LOGBOOKS/`*name*`[0-8]log`
: Log files

`$LOGPATH/`*name*`[0-8]log`
: Log files

SEE ALSO

`log3display`(1M), `log3`(5).

System Administrator's Reference Manual

log3replace(1M)

NAME

log3replace - SET command for Logging V3.0

SYNOPSIS

log3replace [–c *object_class*] [–o *object_name*] [–r "*attribute_value*"...] [–V *sep*]

log3replace [–c *object_class*] –o *object_name*

log3replace –h

DESCRIPTION

The log3replace command can be used to change or delete any data records in Logging V3.0. This deals exclusively with the administration data, such as logbook definitions, event descriptions etc. The logbook entries cannot be modified. The logging status, the logbook cycles and the counter states in Event Management are altered by external factors (start, reset, incoming messages etc.) and likewise cannot be changed by a command call.

The log3replace command is the SET command for Logging V3.0, which can be used to manipulate the data record of any object, to create a new data record and thus a new object and to delete objects.

Notes:

If no object class has been specified, the $CLASS environment variable is evaluated.

If no attribute values are specified (no –r option), the specified object is deleted.

If no object name has been specified, the new object name is taken from the new attribute value settings.

If no object of that name already exists, it will be created by appending the new data record to the relevant file. No error message is issued if a non-existent record is to be replaced, rather a new record is created.

OPTIONS

–c *object_class*

This option determines the object class where changes are to be made. The object class names known at present are as follows [see –H option in log3get(1M)]:

logprotocol
: Definition of the protocol file for the ASCII protocol

logdefault
: Definition of the default logbook

logbooks
: Logbook definitions

logcomponents
: Component definition

logstructures
: Structure descriptions

logevents
: Event descriptions

logreactions
: Reaction descriptions

–o *object_name*

The *object_name* argument determines the data record to be modified, deleted or created or also the object.

In each object class, a different attribute field acts as a unique identifier for an object. In principle, it would also be possible for only a combination of attribute fields to uniquely identify an object. In

extreme cases, only the entire data record with all attribute fields acts as a unique identifier. The objects are identified by the following attribute fields in the object classes:

`logprotocol`
`name`

`logdefault`
`name`

`logbooks`
`name`

`logcomponents`
`componentNumber`

`logstructures`
`<unit>":"number`, where `<unit>` can be the same as "kernel" or "SIlog3"

`logevents`
`unitEventName`

`logreactions`
`unitReactionName`

`-r "`*attribute_value*`"...`
The new attribute values are specified as a list of expressions in the form *attribute_name*=*value*, where *attribute_name* is one of the known attribute names [see `-H` option in `log3get(1M)`] and *value* is the attribute value. Several such expressions are separated by a freely definable separator.

The "=" operator can be surrounded by any number of blanks and tabs. The entire list must be combined to form a single argument with "...".

Several `-r` options can be used.

Example:

```
-r "errorNumber = 1 @ messageTest = HINT"
```

`-V` *sep*
Defines a specific separator. Only a single character is permitted. If this character occurs in the readable ASCII data, it must be masked with "\" in both input and output. The default separator is `@`.

`-h` Usage for this command.

DIAGNOSTICS

The return value is 0 if no error is detected. If the `log3replace` command discovers an error situation, this is reported to standard error output and the return value is not 0.

`1` `"Illegal/less options!"`
Bad call: Incorrect command options or option values were used.

`2` `"Logging V3.0 not installed!"`
The Logging V3.0 home directory cannot be established from the package information. This indicates that Logging V3.0 was not installed.

`3` `"Logging V3.0 not correctly installed!"`
Logging V3.0 has been installed but the necessary commands for reading out data [e.g. `log3eval(1M)`] cannot be found.

`4` `"You are not authorised to administrate Logging V3.0!"`
You are not authorized as a system administrator for Logging V3.0; protected data is therefore not available to you.

`5` `"Unknown object class ".........."!"`
The specified object class does not exist or is not known to the current `log3replace` command.

`6` `"You are not authorised to administrate kernel logging structures!"`
You are not authorized to change the data record.

7 "No delete of default logbook!"

The default logbook cannot be deleted.

8 "No delete of ASCII protocol file!"

The ASCII protocol file cannot be deleted.

9 "Illegal unit (kernel or SIlog3 required)!"

Only the "kernel" or "SIlog3" units can be specified for structure definitions.

10 "You are not authorised to administrate Logging V3.0 event management!"

You are not authorized to change the data record.

11 "Missing or illegal object name $O_OBJECT_NAME!"

Bad call: The object to be changed cannot be established.

12 "Unit (".....") differs from newline (".....")!"

The new data record contains a new unit name, i.e. the object would be moved from one file to another.

13 "Object (".....") differs from newline (".....")!"

The new data record would also change the unique name of the object, i.e. the object would be deleted and entered as a new object. This can only be performed in two steps.

14 "Events/reactions description file "....." is protected!"

The specified event or reaction description files are protected against being changed.

15 "Illegal new record!"

The new data record is incomplete or faulty.

16 "No rename of default logbook or TransView protocol file!"

The names of the default logbook and the ASCII protocol file cannot be modified.

17 "Logbook still in use by $COUNTER component(s)!"

A logbook cannot be deleted if it is still being used. The component assignment must be changed first.

18 "Struct ID $STRUCT illegal for unit "$UNIT"!"

The struct Id is not permitted for the specified unit. The struct Ids are distributed as follows:

kernel	360–499
SIlog3	0-359, 500–

NOTES

The new attribute values are not entered in existing data records, rather all objects with the specified object name are deleted and a new data record with the new attribute values is inserted. This guarantees the uniqueness of the data records and thus the objects following conclusion of the write process. An existing object is thus replaced in this way by a new object with the same name (identifier) but with different attribute values. The position of the object in the file will generally change as a result of this.

If individual attribute values are changed specifically, the data record must be read, the relevant attribute fields must be modified and the data record must be fully restored.

The attribute values are not checked for admissibility, plausibility or completeness.

FILES

LOGPATH=/opt/log3

$LOGPATH/log3admin

$LOGPATH/log3struct

$LOGPATH/default[0-8]log

$LOGBOOKS/*name*[0-8]log

```
$LOGPATH/name[0-8]log
$LOGPATH/protocol.[AB]
$LOGPATH/log3status
$LOGPATH/filters/events.*
$LOGPATH/filters/reactions.*
```

SEE ALSO

log3do(1M), log3eval(1M), log3get(1M), log3status(4), log3(5).

System Administrator's Reference Manual

log3set_pevman(1M)

NAME

System Administrator's Reference Manual

log3str(1M)

NAME

`log3str -` command for converting the `log3struct` administration file for Logging V3.0

SYNOPSIS

`log3str`

DESCRIPTION

The `log3str` command can be used to convert the `log3struct(4)` administration file, which contains structure descriptions for logbook entries for Logging V3.0, to the `log3struct.obj` binary form. It is only if `log3struct(4)` does not contain syntactic errors that the `log3struct.obj` binary version is generated. `log3struct.obj` contains C structures, which are read in by the `log3svr` log server and the `log3display(1M)` and `log3eval(1M)` evaluation commands.

Note:

Not only are the `$LOGPATH/log3struct` and `$ADMPATH/log3struct` files converted to binary form but also any `/etc/default/log3struct` file that may exist. This file contains structure descriptions for messages, which are produced by the kernel and are hence supplied by the operating system software.

OPTIONS

The command has neither options nor usage.

DIAGNOSTICS

The return value is 0 if an error is detected. Standard output is used in this case:

"*n* `STRUCTURE's found.`"

If the `log3str` command discovers an error situation, this is reported to standard error output and the return value is not 0.

`1` "`Location Line:%d Column:%d`"

Syntax error recognized by line and column entry.

NOTES

The file content is only checked for syntactic accuracy, but not for admissibility, plausibility or completeness.

FILES

`$LOGPATH/log3struct`
`$ADMPATH/log3struct`
`$LOGPATH/log3struct.obj`
`$ADMPATH/log3struct.obj`
`/etc/default/log3struct`

SEE ALSO

`log3display(1M)`, `log3eval(1M)`, `log3struct(4)`, `log3(5)`.

System Administrator's Reference Manual

logins(1M)

NAME

`logins -` list user and system login information

SYNOPSIS

`logins [ –dmopstuxab ] [ –g` *groups* `] [ –l` *logins* `]`

DESCRIPTION

This command displays information on user and system logins. The content of the output is controlled by the command options and can include the following: user or system login, user id number, `/etc/passwd` account field value (user name or other information), primary group name, primary group id, multiple group names, multiple group ids, home directory, login shell, and four password aging parameters. The default information is the following: login name, user id, primary group name, primary group id and the account field value from `/etc/passwd`. Output is sorted by user id, displaying system logins followed by user logins.

OPTIONS

`–d` Selects logins with duplicate uids.

`–m` Displays multiple group membership information.

`–o` Formats output into one line of colon-separated fields.

`–p` Selects logins with no passwords.

`–s` Selects all system logins.

`–t` Sorts output by login instead of by uid.

`–u` Selects all user logins.

`–x` Prints an extended set of information about each selected user. The extended information includes home directory, login shell and password aging information, each displayed on a separate line. The password information consists of password status (PS for passworded, NP for no password or LK for locked). If the login is passworded, status is followed by the date the password was last changed, the number of days required between changes, and the number of days allowed before a change is required. The password aging information shows the time interval that the user will receive a password expiration warning message (when logging on) before the password expires.

`–a` Adds two password expiration fields to the display. The fields show how many days a password can remain unused before it automatically becomes inactive and the date that the password will expire.

`–g` Selects all users belonging to group, sorted by login. Multiple groups can be specified as a comma-separated list.

`–l` Selects the requested login. Multiple logins can be specified as a comma-separated list.

`–b` Prints the user's audit event mask. This option is only valid if the auditing subsystem is installed.

LOCALE

The `LC_MESSAGES` environment variable governs the language in which message texts are displayed.

If `LC_MESSAGES` is undefined or is defined as the null string, it defaults to the value of `LANG`. If `LANG` is likewise undefined or null, the system acts as if it were not internationalized.

If any of the locale variables has an invalid value, the system acts as if none of the variables was set.

The `LC_ALL` environment variable governs the entire locale. `LC_ALL` takes precedence over all the other environment variables that affect internationalization.

NOTES

Options may be used together. If so, any login matching any criteria will be displayed. When the `–l` and `–g`

options are combined, a user will only be listed once, even if they belong to more than one of the selected groups.

System Administrator's Reference Manual

logmsg(1M)
RM400 only

NAME

`logmsg -` log system startup messages

SYNOPSIS

`/sbin/logmsg` *tag priority* [*message*]

`/sbin/logmsg –copy`

DESCRIPTION

`logmsg` is used for logging system messages during the startup and shutdown phases where the `syslogd` daemon is not yet or is no longer active. Messages that are passed on to `logmsg` are stored in a temporary file `/startup_log.tmp`, and are later passed on to the `syslogd` daemon explicitly using the "`/sbin/logmsg –copy`" command.

Two variants of `logmsg` can be used:

`/sbin/logmsg` *tag priority message*

or

command | `/sbin/logmsg` *tag priority*

Each message is assigned a *tag* and a priority. The tag, priority, and message are stored temporarily with a time stamp in the file `/startup_log.tmp`.

ARGUMENTS

tag Identifies the sender of the message and should be the name of the sending program (e.g. "$0").

priority

The values of *priority* correspond to those defined in `syslog.conf(4)` for "level", e.g. `debug`, `info`, `notice`. Messages with a higher priority than "info" are passed directly on to `stdout`, but are later passed on to the `syslogd` daemon with priority `user.info`. Messages with a higher priority than `notice` are appended with the word "WARNING" or "ERROR".

message

Any text. Interpretation of messages with metacharacters (e.g. `\n`) corresponds to that of the `echo(1)` command.

If no specification is made for *message*, `stdin` input is used.

`–copy`

After the `syslogd` daemon has been started, all temporarily stored messages are to be passed on to the `syslogd` daemon using "`/sbin/logmsg –copy`". The `/startup_log.tmp` file is deleted after successful transfer.

All messages are logged to the *facility* parameter `user` and the *level* parameter `debug` or `notice` by `syslog(3C)` with the tag `` `basename` *tag*``.

USAGE

`logmsg` can be used in all `rc`*X* scripts. Logging of all startup and shutdown messages is made possible as a result of the following modifications:

1) In the file `/etc/inittab`:

Replace all calls for

`/sbin/rc`*X*

with

`/sbin/rc`*X*`_log`

(*X* stands for `0`, `1`, `2`, `3`, `6`, `S`).

After the line "r6:...", and before the line "co:...", insert:

```
sl:23:wait:/sbin/logmsg -copy >/dev/console 2>&1
```

2) In `/sbin/rc2` and `/sbin/rc3`:

As the last line, insert:

```
echo "ExIt LoGmSg"
```

This string from `echo` is interpreted by `logmsg` and terminates the program called in a pipe by `/sbin/rc`*X*`_log`.

In a standard configuration, all startup and shutdown messages are logged in `/var/adm/log/messages`.

FILES

`/startup_log.tmp`
`/var/adm/log/messages`

SEE ALSO

`syslogd`(1M), `syslog`(3C), `syslog.conf`(4).

System Administrator's Reference Manual

lpadmin(1M)

NAME

`lpadmin -` configure the LP print service

SYNOPSIS

`lpadmin –p` *printer options*

`lpadmin –x` *dest*

`lpadmin –d` [*dest*]

`lpadmin –S` *print-wheel* `–A` *alert-type* [`–W` *minutes*] [`–Q` *requests*]

DESCRIPTION

`lpadmin` configures the LP print service by defining printers and devices. It is used to add and change printers, to remove printers from the service, to set or change the system default destination, to define alerts for printer faults, and to mount print wheels.

Adding or Changing a Printer

The first form of the `lpadmin` command (`lpadmin –p` *printer options*) is used to configure a new printer or to change the configuration of an existing printer. The following *options* may appear in any order.

`–A` *alert-type* [`–W` *minutes*]

The `–A` option is used to define an alert to inform the administrator when a printer fault is detected, and periodically thereafter, until the printer fault is cleared by the administrator. The *alert-types* are:

`mail` Sends the alert message via mail [see `mail(1)`] to the administrator.

`write` Writes the message to the terminal on which the administrator is logged in. If the administrator is logged in on several terminals, one is chosen arbitrarily.

`quiet` Messages are not sent for the current condition. An administrator can use this option to temporarily stop receiving further messages about a known problem. Once the fault has been cleared and printing resumes, messages will again be sent when another fault occurs with the printer.

`none` No messages are sent; any existing alert definition for the printer will be removed. No alert will be sent when the printer faults until a different alert-type (except `quiet`) is used.

shell-command

Runs the *shell-command* each time the alert needs to be sent. The shell command should expect the message in standard input. If there are blanks embedded in the command, enclose the command in quotes. Note that the `mail` and `write` values for this option are equivalent to the values `mail` *user-name* and `write` *user-name* respectively, where *user-name* is the current name for the administrator. This will be the login name of the person submitting this command unless he or she has used the `su` command to change to another user ID. If the `su` command has been used to change the user ID, then the *user-name* for the new ID is used.

`list` Displays the type of the alert for the printer fault. No change is made to the alert.

The message sent appears as follows:

`The printer` *printer* has stopped printing for the reason given below. Fix the problem and bring the printer back on line. Printing has stopped, but will be restarted in a few minutes; issue an enable command if you want to restart sooner. Unless someone issues a change request

lp –i *request-id* `–P ...`

```
to change the page list to print, the current request will be reprinted
from the beginning.
The reason(s) it stopped (multiple reasons indicate reprinted attempts):
```

reason

The LP print service can detect printer faults only through an adequate fast filter and only when the standard interface program or a suitable customized interface program is used. Furthermore, the level of recovery after a fault depends on the capabilities of the filter.

If the *printer* is `all`, the alerting defined in this command applies to all existing printers.

If the `-W` option is not used to arrange fault alerting for *printer*, the default procedure is to mail one message to the administrator of *printer* per fault. This is equivalent to specifying `-W once` or `-W 0`. If *minutes* is a number greater than zero, an alert will be sent at intervals specified by *minutes*.

`-c` *class*

Inserts *printer* into the specified *class*. *class* will be created if it does not already exist.

`-D` *comment*

Saves this *comment* for display whenever a user asks for a full description of *printer* [see `lpstat(1)`]. The LP print service does not interpret this comment.

`-e` *$printer sub 1$*

Copies the interface program of an existing *$printer sub 1$* to be the interface program for *printer*. (Options `-i` and `-m` may not be specified with this option.)

`-F` *fault-recovery*

Specifies the recovery to be used for any print request that is stopped because of a printer fault, according to the value of *fault-recovery*:

`continue`

Continues printing from the top of the page where printing stopped. This requires a filter to wait for the fault to clear before continuing automatically.

`beginning`

Starts printing the request again from the beginning.

`wait` Disables printing on *printer* and waits for the administrator or a user to enable printing again.

During the wait the administrator or the user who submitted the stopped print request can issue a change request that specifies where printing should resume. (See the `-i` option of the `lp` command.) If no change request is made before printing is enabled, printing will resume at the top of the page where stopped, if the filter allows; otherwise, the request will be printed from the beginning.

`-f allow:`*form-list*

`-f deny:`*form-list*

Allows or denies the forms in *form-list* to be printed on *printer*. By default no forms are allowed on a new printer.

For each printer, the LP print service keeps two lists of forms: an "allow-list" of forms that may be used with the printer, and a "deny-list" of forms that may not be used with the printer. With the `-f allow` option, the forms listed are added to the allow-list and removed from the deny-list. With the `-f deny` option, the forms listed are added to the deny-list and removed from the allow-list.

If the allow-list is not empty, only the forms in the list may be used on the printer, regardless of the contents of the deny-list. If the allow-list is empty, but the deny-list is not, the forms in the deny-list may not be used with the printer. All forms can be excluded from a printer by specifying `-f deny:all`. All forms can be used on a printer (provided the printer can handle all the characteristics of each form) by specifying `-f allow:all`.

The LP print service uses this information as a set of guidelines for determining where a form can be mounted. Administrators, however, are not restricted from mounting a form on any printer. If mounting a form on a particular printer is in disagreement with the information in the allow-list or deny-list, the administrator is warned but the mount is accepted. Nonetheless, if a user attempts to issue a print or change request for a form and printer combination that is in disagreement with the information, the request is accepted only if the form is currently mounted on the printer. If the form is later unmounted before the request can print, the request is canceled and the user is notified by mail.

If the administrator tries to specify a form as acceptable for use on a printer that doesn't have the capabilities needed by the form, the command is rejected.

Note the other use of `-f`, with the `-M` option, below.

`-h` Indicates that the device associated with the printer is hardwired. If neither of the mutually exclusive options, `-h` and `-l`, is specified, this option is assumed.

`-I` *content-type-list*
Allows *printer* to handle print requests with the content types listed in a *content-type-list*. If the list includes names of more than one type, the names must be separated by commas or blank spaces. (If they are separated by blank spaces, the entire list must be enclosed in double quotes.)

The type `simple` is recognized as the default content type for files in the Reliant UNIX system. A `simple` type of file is a data stream containing only printable ASCII characters and the following control characters.

Control Character	Octal Value	Meaning
backspace	10_8	move back one character, except at beginning of line
tab	11_8	move to next tab stop
line feed (newline)	12_8	move to beginning of next line
form feed	14_8	move to beginning of next page
carriage return	15_8	move to beginning of current line

To prevent the print service from considering `simple` a valid type for the printer, specify either an explicit value (such as the printer type) in the *content-type-list*, or an empty list. If you do want `simple` included along with other types, you must include `simple` in the *content-type-list*.

Except for `simple`, each *content-type* name is freely determined by the administrator. If the printer type is specified by the `-T` option, then the printer type is implicitly considered to be also a valid content type.

`-i` *interface*
Establishes a new interface program for *printer*. *interface* is the pathname of the new program. (The `-e` and `-m` options may not be specified with this option.)

`-l` Indicates that the device associated with *printer* is a login terminal. The LP scheduler (`lpsched`) disables all login terminals automatically each time it is started. (The `-h` option may not be specified with this option.)

`-M -f` *form-name* [`-a` [`-o filebreak`]]
Mounts the form *form-name* on *printer*. Print requests that need the pre-printed form *form-name* are printed on *printer*. If more than one printer has the form mounted and the user has specified `any` (`-d` option of `lp` command) as the printer destination, the print request will be printed on the printer that also meets the other needs of the request.

The page length and width, character and line pitches needed by the form are compared with those allowed for the printer, by checking the capabilities in the `terminfo` database for the type of printer. If the form requires attributes that are not available with the printer, the administrator is warned but the mount is accepted. If the form lists a print wheel as mandatory, but the print wheel mounted on the printer is different, the administrator is also warned but the mount is accepted.

The `-a` option prints an alignment pattern, preceded by the same initialization of the physical printer that precedes a normal print request, except that no banner page is printed. Printing is assumed to start at the top of the first page of the form. After the pattern is printed, the administrator can adjust the mounted form in the printer and press return for another alignment pattern (no initialization this time), and can continue printing alignment patterns as desired. The administrator can quit the printing of

alignment patterns by typing `q`.

If the `-o filebreak` option is given, a formfeed is inserted between each copy of the alignment pattern. By default, the alignment pattern is assumed to correctly fill a form, so no formfeed is added.

A form is "unmounted" either by mounting a new form in its place or by using the `-f none` option. By default, a new printer has no form mounted.

Note the other use of `-f` without the `-M` option above.

`-M -S` *print-wheel*

Mounts the *print-wheel* on *printer*. Print requests that need the *print-wheel* will be printed on *printer*. If more than one printer has *print-wheel* mounted and the user has specified `any` (with the `-d` option of the `lp` command) as the printer destination, then the print request will be printed on the one printer that also meets the other needs of the request.

If the *print-wheel* is not listed as acceptable for the printer, the administrator is warned but the mount is accepted. If the printer does not take print wheels, the command is rejected.

A print wheel is "unmounted" either by mounting a new print wheel in its place or by using the option `-S none`. By default, a new printer has no print wheel mounted.

Note the other uses of the `-S` option without the `-M` option described below.

`-m` *model*

Selects *model* interface program, provided with the LP print service, for the printer. (Options `-e` and `-i` may not be specified with this option.)

`-o` *printing-option*

Each `-o` option in the list below is the default given to an interface program if the option is not taken from a preprinted form description or is not explicitly given by the user submitting a request [see `lp`(1)]. The only `-o` options that can have defaults defined are listed below.

`length=`
scaled-decimal-number

`width=`
scaled-decimal-number

`cpi=` *scaled-decimal-number*

`lpi=` *scaled-decimal-number*

`stty=` *'stty-option-list'*

The term "scaled-decimal-number" refers to a non-negative number used to indicate a unit of size. The type of unit is shown by a "trailing" letter attached to the number. Three types of scaled decimal numbers can be used with the LP print service: numbers that show sizes in centimeters (marked with a trailing `c`); numbers that show sizes in inches (marked with a trailing `i`); and numbers that show sizes in units appropriate to use (without a trailing letter), that is, lines, characters, lines per inch, or characters per inch.

The first four default option values must agree with the capabilities of the type of physical printer, as defined in the `terminfo` database for the printer type. If they do not, the command is rejected.

The *stty-option-list* is not checked for allowed values, but is passed directly to the `stty` program by the standard interface program. Any error messages produced by `stty` when a request is processed (by the standard interface program) are mailed to the user submitting the request.

For each printing option not specified, the defaults for the following attributes are defined in the `terminfo` entry for the specified printer type.

`length`

`width`

`cpi`

`lpi`

```
The default for stty is
stty='9600 cs8 -cstopb -parenb ixon
       -ixany opost -olcuc onlcr -ocrnl -onocr
       -onlret -ofill nl0 cr0 tab0 bs0 vt0 ff0'
```

You can set any of the `-o` options to the default values (which vary for different types of printers), by typing them without assigned values, as follows:

```
length=
width=
cpi=
lpi=
stty=
```

`-o nobanner`
Allows a user to submit a print request specifying that no banner page be printed.

`-o banner`
Forces a banner page to be printed with every print request, even when a user asks for no banner page. This is the default; you must specify `-o nobanner` if you want to allow users to be able to specify `-o nobanner` with the `lp` command.

`-r` *class*
Removes *printer* from the specified *class*. If *printer* is the last member of *class*, then *class* will be removed.

`-S` *list*
Allows either the print wheels or aliases for character sets named in *list* to be used on the printer.

If the printer is a type that takes print wheels, then *list* is a comma or space separated list of print wheel names. (Enclose the list with quotes if it contains blanks.) These will be the only print wheels considered mountable on the printer. (You can always force a different print wheel to be mounted, however.) Until the option is used to specify a list, no print wheels will be considered mountable on the printer, and print requests that ask for a particular print wheel with this printer will be rejected.

If the printer is a type that has selectable character sets, then *list* is a comma or blank separated list of character set name "mappings" or aliases. (Enclose the list with quotes if it contains blanks.) Each "mapping" is of the form

known-name=alias

The *known-name* is a character set number preceded by `cs` (such as `cs3` for character set three) or a character set name from the `terminfo` database entry `csnm` [see `terminfo(4)`]. If this option is not used to specify a list, only the names already known from the Terminfo database or numbers with a prefix of `cs` will be acceptable for the printer.

If *list* is the word `none`, any existing print wheel lists or character set aliases will be removed.

Note the other uses of the `-S` with the `-M` option described above.

`-s` *system-name* [!*printer-name*]
Makes a remote printer (one that must be accessed through another system) accessible to users on your system. *System-name* is the name of the remote system on which the remote printer is located; it must be listed in the systems table (`/etc/lp/Systems`). *Printer-name* is the name used on the remote system for that printer. For example, if you want to access *$printer sub 1$* on *$system sub 1$* and you want it called *$printer sub 2$* on your system, enter `-p` *$printer sub 2$* `-s` *$system sub 1$*!*$printer sub 1$*

`-T` *printer-type-list*
Identifies the printer as being of one or more *printer-type*s. Each *printer-type* is used to extract data from the `terminfo` database; this information is used to initialize the printer before printing each user's request. Some filters may also use a *printer-type* to convert content for the printer. If this option is not

used, the default *printer-type* will be `unknown`; no information will be extracted from `terminfo` so each user request will be printed without first initializing the printer. Also, this option must be used if the following are to work: `–o cpi`, `–o lpi`, `–o width`, and `–o length` options of the `lpadmin` and `lp` commands, and the `–S` and `–f` options of the `lpadmin` command.

If the *printer-type-list* contains more than one type, then the *content-type-list* of the `–I` option must either be specified as `simple`, as empty (`–I ""`), or not specified at all.

`–u allow:`*login-ID-list*
`–u deny:`*login-ID-list*

Allows or denies the users in *login-ID-list* access to the printer. By default all users are allowed on a new printer. The *login-ID-list* argument may include any or all of the following constructs:

login-ID
auser on any system

system-name`!`*login-ID*
auser on system *system-name*

system-name`!all`
all users on system *system-name*

`all!`*login-ID*
auser on all systems

`all` all users on all systems

The LP print service keeps two lists of users for each printer: an "allow-list" of people allowed to use the printer and a "deny-list" of people denied access to the printer. With the `–u allow` option, the users listed are added to the allow-list and removed from the deny-list. With the `–u deny` option, the users listed are added to the deny-list and removed from the allow-list.

If the allow-list is not empty, only the users in the list may use the printer, regardless of the contents of the deny-list. If the allow-list is empty, but the deny-list is not, the users in the deny-list may not use the printer. All users can be denied access to the printer by specifying `–u deny:all`. All users may use the printer by specifying `–u allow:all`.

`–U` *dial-info*

Allows your print service to access a remote printer. (It does not enable your print service to access a remote printer service.) Specifically, `–U` assigns the "dialing" information *dial-info* to the printer. *Dial-info* is used with the `dial` routine to call the printer. Any network connection supported by the Basic Networking Utilities will work. *Dial-info* can be either a phone number for a modem connection, or a system name for other kinds of connections. Or, if `–U direct` is given, no dialing will take place, because the name `direct` is reserved for a printer that is directly connected. If a system name is given, it is used to search for connection details from the file `/etc/uucp/Systems` or related files. The Basic Networking Utilities are required to support this option. By default, `–U direct` is assumed.

`–v` *device*

Associates a *device* with *printer*. *device* is the path name of a file that is writable by `lp`. Note that the same *device* can be associated with more than one printer.

Restrictions

When creating a new printer, one of three options (`–v`, `–U`, or `–s\`) must be supplied. In addition, only one of the following may be supplied: `–e`, `–i`, or `–m`; if none of these three options is supplied, the model standard is used. The `–h` and `–l` options are mutually exclusive. Printer and class names may be no longer than 14 characters and must consist entirely of the characters `A-Z`, `a-z`, `0-9` and `_` (underscore). If `–s` is specified, the following options are invalid: `–A`, `–e`, `–F`, `–h`, `–i`, `–l`, `–M`, `–m`, `–o`, `–U`, `–v`, and `–W`.

Removing a Printer Destination

The `–x` *dest* option removes the destination *dest* (a printer or a class), from the LP print service. If *dest* is a

printer and is the only member of a class, then the class will be deleted, too. If *dest* is `all`, all printers and classes are removed. No other *options* are allowed with `-x`.

Setting/Changing the System Default Destination

The `-d` [*dest*] option makes *dest*, an existing printer or class, the new system default destination. If *dest* is not supplied, then there is no system default destination. No other *options* are allowed with `-d`.

Setting an Alert for a Print Wheel

`-S` *print-wheel* `-A` *alert-type* [`-W` *minutes*] [`-Q` *requests*]

The `-S` *print-wheel* option is used with the `-A`*alert-type* option to define an alert to mount the print wheel when there are jobs queued for it. If this command is not used to arrange alerting for a print wheel, no alert will be sent for the print wheel. Note the other use of `-A`, with the `-p` option, above.

The *alert-types* are:

`mail` Sends the alert message via the `mail` command to the administrator.

`write` Writes the message, via the `write` command, to the terminal on which the administrator is logged in. If the administrator is logged in on several terminals, one is arbitrarily chosen.

`quiet` Messages are not sent for the current condition. An administrator can use this option to temporarily stop receiving further messages about a known problem. Once the *print-wheel* has been mounted and subsequently unmounted, messages will again be sent when the number of print requests reaches the threshold specified by the `-Q` option.

`none` Messages are not sent until the `-A` option is given again with a different *alert-type* (other than `quiet`).

shell-command
Runs the *shell-command* each time the alert needs to be sent. The shell command should expect the message in standard input. If there are blanks embedded in the command, enclose the command in quotes. Note that the `mail` and `write` values for this option are equivalent to the values `mail` *user-name* and `write` *user-name* respectively, where *user-name* is the current name for the administrator. This will be the login name of the person submitting this command unless he or she has used the `su` command to change to another user ID. If the `su` command has been used to change the user ID, then the *user-name* for the new ID is used.

`list` Displays the type of the alert for the print wheel on standard output. No change is made to the alert.

The message sent appears as follows:

`The print wheel` *print-wheel* needs to be mounted on the printer(s):

printer `(`*$integer sub 1$* `requests)`*$integer sub 2$* `print requests await this print wheel.`

The printers listed are those that the administrator had earlier specified were candidates for this print wheel. The number *$integer sub 1$* listed next to each printer is the number of requests eligible for the printer. The number *$integer sub 2$* shown after the printer list is the total number of requests awaiting the print wheel. It will be less than the sum of the other numbers if some requests can be handled by more than one printer.

If the *print-wheel* is `all`, the alerting defined in this command applies to all print wheels already defined to have an alert.

If the `-W` option is not given, the default procedure is that only one message will be sent per need to mount the print wheel. Not specifying the `-W` option is equivalent to specifying `-W once` or `-W0`. If *minutes* is a number greater than zero, an alert will be sent at intervals specified by *minutes*.

If the `-Q` option is also given, the alert will be sent when a certain number (specified by the argument *requests*) of print requests that need the print wheel are waiting. If the `-Q` option is not given, or *requests* is 1 or the word `any` (which are both the default), a message is sent as soon as anyone submits a print request for the print wheel when it is not mounted.

FILES

```
/var/spool/lp/*
/etc/lp
```

SEE ALSO

enable(1), lp(1), lpstat(1), stty(1), accept(1M), lpsched(1M), lpsystem(1M), dial(3C), terminfo(4).

System Administrator's Reference Manual

lpc(1M)
BSD Compatibility Package

NAME

`lpc` - line printer control program

SYNOPSIS

`/usr/ucb/lpc` [*command* [*parameter* ...]]

DESCRIPTION

`lpc` controls the operation of the printer, or of multiple printers. `lpc` commands can be used to start or stop a printer, disable or enable a printer's spooling queue, rearrange the order of jobs in a queue, or display the status of each printer – along with its spooling queue and printer daemon.

With no arguments, `lpc` runs interactively, prompting with `lpc>`. If arguments are supplied, `lpc` interprets the first as a *command* to execute; each subsequent argument is taken as a *parameter* for that command. The standard input can be redirected so that `lpc` reads commands from a file.

Commands may be abbreviated to an unambiguous substring. The *printer* parameter is specified just by the name of the printer (for example, `lw`), not as you would specify it to `lpr(1)` or `lpq(1)` (not `-Plw`).

`?` [*command*]...
`help` [*command*]...
: Display a short description of each command specified in the argument list, or, if no arguments are given, a list of the recognized commands.

`abort` [`all` | [*printer* ...]]
: Terminate an active spooling daemon on the local host immediately and then disable printing [preventing new daemons from being started by `lpr(1)`] for the specified printers. The `abort` command can only be used by the privileged user.

`clean` [`all` | [*printer* ...]]
: Remove all files created in the spool directory by the daemon from the specified printer queue(s) on the local machine. The `clean` command can only be used by the privileged user.

`disable` [`all` | [*printer* ...]]
: Turn the specified printer queues off. This prevents new printer jobs from being entered into the queue by `lpr(1)` . The `disable` command can only be used by the privileged user.

`down` [`all` | [*printer* ...]] [*message*]
: Turn the specified printer queue off, disable printing and put *message* in the printer status file. The message does not need to be quoted, the remaining arguments are treated like `echo(1)` . This is normally used to take a printer down and let others know why [`lpq(1)` indicates that the printer is down, as does the `status` command].

`enable` [`all` | [*printer* ...]]
: Enable spooling on the local queue for the listed printers, so that `lpr(1)` can put new jobs in the spool queue. The `enable` command can only be used by the privileged user.

`exit`
`quit`
: Exit from `lpc`.

`restart` [`all` | [*printer* ...]]
: Attempt to start a new printer daemon. This is useful when some abnormal condition causes the daemon to die unexpectedly leaving jobs in the queue. This command can be run by any user.

`start` [`all` | [*printer* ...]]
: Enable printing and start a spooling daemon for the listed printers. The `start` command can only be used by the privileged user.

`status` [`all` | [*printer* ...]]
: Display the status of daemons and queues on the local machine. This command can be run by any user.

`stop` [`all` | [*printer* ...]]
: Stop a spooling daemon after the current job completes and disable printing. The `stop` command can only be used by the privileged user.

`topq` *printer* [*job#* ...] [*user* ...]
: Move the print job(s) specified by *job#* or those job(s) belonging to *user* to the top (head) of the printer queue. The `topq` command can only be used by the privileged user.

`up` [`all` | [*printer* ...]]
: Enable everything and start a new printer daemon. Undoes the effects of `down`.

DIAGNOSTICS

`?Ambiguous command`
: The abbreviation you typed matches more than one command.

`?Invalid command`
: You typed a command or abbreviation that was not recognized.

`?Privileged command`
: You used a command can be executed only by the privileged user.

`lpc:` *printer*: unknown printer to the print service
: The `printer` was not found in the System V LP database. Usually this is a typing mistake; however, it may indicate that the printer does not exist on the system. Use `lptstat –p` to find the reason.

`lpc: error on opening queue to spooler`
: The connection to `lpsched` on the local machine failed. This usually means the printer server started at boot time has died or is hung. Check if the printer spooler daemon `/usr/lib/lp/lpsched` is running.

`lpc: Can't send message to LP print service`
`lpc: Can't receive message from LP print service`
: These indicate that the LP print service has been stopped. Get help from the system administrator.

`lpc: Received unexpected message from LP print service`
: It is likely there is an error in this software. Get help from system administrator.

FILES

`/var/spool/lp/*`

`/var/spool/lp/system/pstatus`

SEE ALSO

`echo`(1), `lpq`(1), `lpr`(1), `lprm`(1), `lpsched`(1M).

System Administrator's Reference Manual

lpfilter(1M)

NAME

`lpfilter -` administer filters used with the LP print service

SYNOPSIS

`lpfilter –f` *filter-name* `–F` *path-name*

`lpfilter –f` *filter-name* `–`

`lpfilter –f` *filter-name* `–i`

`lpfilter –f` *filter-name* `–x`

`lpfilter –f` *filter-name* `–l`

DESCRIPTION

The `lpfilter` command is used to add, change, delete, and list a filter used with the LP print service. These filters are used to convert the content type of a file to a content type acceptable to a printer. One of the following options must be used with the `lpfilter` command: `–F`*path-name* (or "`–`" for standard input) to add or change a filter; `–i` to reset an original filter to its factory setting; `–x` to delete a filter; or `–l` to list a filter description.

The argument `all` can be used instead of a *filter-name* with any of these options. When `all` is specified with the `–F` or "`–`" option, the requested change is made to all filters. Using `all` with the `–i` option has the effect of restoring to their original settings all filters for which predefined settings were initially available. Using the `all` argument with the `–x` option results in all filters being deleted, and using it with the `–l` option produces a list of all filters.

Adding or Changing a Filter

The filter named in the `–f` option is added to the filter table. If the filter already exists, its description is changed to reflect the new information in the input.

The filter description is taken from the *path-name* if the `–F` option is given, or from standard input if the `–` option is given. One of the two must be given to define or change a filter. If the filter named is one originally delivered with the LP print service, the `–i` option will restore the original filter description. When an existing filter is changed with the `–F` or "`–`" option, items that are not specified in the new information are left as they were. When a new filter is added with this command, unspecified items are given default values (see below).

Filters are used to convert the content of a request into a data stream acceptable to a printer. For a given print request, the LP print service will know the following: the type of content in the request, the name of the printer, the type of the printer, the types of content acceptable to the printer, and the modes of printing asked for by the originator of the request. It will use this information to find a filter or a pipeline of filters that will convert the content into a type acceptable to the printer. Below is a list of items that provide input to this command, and a description of each item. All lists are separated by commas or spaces.

`Input types:` *content-type-list*
`Output types:` *content-type-list*
`Printer types:` *printer-type-list*
`Printers:` *printer-list*
`Filter type:` *filter-type*
`Command:` *shell-command*
`Options:` *template-list*

`Input types`
: This gives the types of content that can be accepted by the filter. (The default is `any`.)

`Output types`
: This gives the types of content that the filter can produce from any of the input content types. (The

default is `any`.)

`Printer types`
: This gives the type of printers for which the filter can be used. The LP print service will restrict the use of the filter to these types of printers. (The default is `any`.)

`Printers`
: This gives the names of the printers for which the filter can be used. The LP print service will restrict the use of the filter to just the printers named. (The default is `any`.)

`Filter type`
: This marks the filter as a `slow` filter or a `fast` filter. Slow filters are generally those that take a long time to convert their input. They are run unconnected to a printer, to keep the printers from being tied up while the filter is running. If a listed printer is on a remote system, the filter type for it must have the value `slow`. Fast filters are generally those that convert their input quickly, or those that must be connected to the printer when run. These will be given to the interface program to run connected to the physical printer.

`Command`
: This specifies the program to run to invoke the filter. The full program pathname as well as fixed options must be included in the *shell-command*; additional options are constructed, based on the characteristics of each print request and on the `Options` field. A command must be given for each filter.

  The command must accept a data stream as standard input and produce the converted data stream on its standard output. This allows filter pipelines to be constructed to convert data not handled by a single filter.

`Options`
: This is a comma separated list of templates used by the LP print service to construct options to the filter from the characteristics of each print request listed in the table later.

  In general, each template is of the following form:

  keyword pattern = replacement

  The *keyword* names the characteristic that the template attempts to map into a filter specific option; each valid *keyword* is listed in the table below. A *pattern* is one of the following: a literal pattern of one of the forms listed in the table, a single asterisk (`*`), or a regular expression. If *pattern* matches the value of the '%n 5 characteristic, the template fits and is used to generate a filter specific option. The *replacement* is what will be used as the option.

Regular expressions are the same as those found in the `ed(1)` or `vi(1)` commands. This includes the `\(...\)` and `\`*n* constructions, which can be used to extract portions of the *pattern* for copying into the *replacement*, and the `&`, which can be used to copy the entire *pattern* into the *replacement*.

The *replacement* can also contain a `*`; it too, is replaced with the entire *pattern*, just like the `&` of `ed(1)` .

`lp` Option	Characteristic	*keyword*	Possible *pattern*s
`–T`	Content type (input)	`INPUT`	*content-type*
`N/A`	Content type (output)	`OUTPUT`	*content-type*
`N/A`	Printer type	`TERM`	*printer-type*
`–d`	Printer name	`PRINTER`	*printer-name*
`–f`, –o cpi=	Character pitch	`CPI`	*integer*
`–f`, –o lpi=	Line pitch	`LPI`	*integer*
`–f`, –o length=	Page length	`LENGTH`	*integer*
`–f`, –o width=	Page width	`WIDTH`	*integer*
`–P`	Pages to print	`PAGES`	*page-list*

–S	Character set	CHARSET	*character-set-name*
	Print wheel	CHARSET	*print-wheel-name*
–f	Form name	FORM	*form-name*
–y	Modes	MODES	*mode*
–n	Number of copies	COPIES	*integer*

For example, the template

```
MODES landscape = –l
```

shows that if a print request is submitted with the –y landscape option, the filter will be given the option –l. As another example, the template

```
TERM * = –T *
```

shows that the filter will be given the option –T *printer-type* for whichever *printer-type* is associated with a print request using the filter.

As a last example, consider the template

```
MODES prwidth\=\(.*\) = –w\1
```

Suppose a user gives the command

```
lp –y prwidth=10
```

From the table above, the LP print service determines that the –y option is handled by a MODES template. The MODES template here works because the *pattern* prwidth\=\(.*\) matches the prwidth=10 given by the user. The *replacement* –w\1 causes the LP print service to generate the filter option –w10.

If necessary, the LP print service will construct a filter pipeline by concatenating several filters to handle the user's file and all the print options. (See sh(1) for a description of a pipeline.) If the print service constructs a filter pipeline, the INPUT and OUTPUT values used for each filter in the pipeline are the types of the input and output for that filter, not for the entire pipeline.

Deleting a Filter

The –x option is used to delete the filter specified in *filter-name* from the LP filter table.

Listing a Filter Description

The –l option is used to list the description of the filter named in *filter-name*. If the command is successful, the following message is sent to standard output:

```
Input types: content-type-list□
Output types: content-type-list□
Printer types: printer-type-list□
Printers: printer-list□
Filter type: filter-type□
Command: shell-command□
Options: template-list
```

If the command fails, an error message is sent to standard error.

SEE ALSO

lp(1), lpadmin(1M).

System Administrator's Reference Manual

lpforms(1M)

NAME

`lpforms -` administer forms used with the LP print service

SYNOPSIS

`lpforms –f` *form_name options*

`lpforms –f` *form_name* `–A` *alert_type* [`–Q` *minutes*] [`–W` *requests*]

DESCRIPTION

The `lpforms` command is used to administer the use of preprinted forms, such as company letterhead paper, with the LP print service. A form is specified by its *form_name*. Users may specify a form when submitting a print request [see `lp(1)`]. The argument `all` can be used instead of *form_name* with either of the command lines shown above. The first command line allows the administrator to add, change, and delete forms, to list the attributes of an existing form, and to allow and deny users access to particular forms. The second command line is used to establish the method by which the administrator is alerted that the form *form_name* must be mounted on a printer.

OPTIONS

With the first `lpforms` command line, one of the following options must be used:

`–F` *pathname*
: Adds or changes a form *form_name*, as specified by the information in *pathname*.

`–`
: Adds or changes a form *form_name*, as specified by the information from standard input.

`–x`
: Deletes a form *form_name* (this option must be used separately; it may not be used with any other option).

`–l`
: Lists the attributes of a form *form_name*.

Adding or Changing a Form

The `–F` *pathname* option is used to add a new form, *form_name*, to the LP print service, or to change the attributes of an existing form. The form description is taken from *pathname* if the `–F` option is given, or from the standard input if the "–" option is used. One of these two options must be used to define or change a form. *pathname* is the path name of a file that contains all or any subset of the following information about the form.

`Page length:` $scaled\text{-}decimal\text{-}number_1$

`Page width:` $scaled\text{-}decimal\text{-}number_2$

`Number of pages:` *integer*
`Line pitch:` $scaled\text{-}decimal\text{-}number_3$

`Character pitch:` $scaled\text{-}decimal\text{-}number_4$

`Character set choice:` *character-set/print-wheel*
[`mandatory`]
`Ribbon color:` *ribbon-color*
`Comment :` *comment*
`Alignment pattern:` [*content-type*] *content*

The term "scaled-decimal-number" refers to a non-negative number used to indicate a unit of size. The type of

unit is shown by a "trailing" letter attached to the number. Three types of scaled decimal numbers can be used with the LP print service: numbers that show sizes in centimeters (marked with a trailing `c`); numbers that show sizes in inches (marked with a trailing `i`); and numbers that show sizes in units appropriate to use (without a trailing letter), that is, lines, characters, lines per inch, or characters per inch.

Except for the last two lines, the above lines may appear in any order. The `Comment:` and *comment* items must appear in consecutive order but may appear before the other items, and the `Alignment pattern:` and the *content* items must appear in consecutive order at the end of the file. Also, the *comment* item may not contain a line that begins with any of the key phrases above, unless the key phrase is preceded with a "`>`" sign. Any leading "`>`" sign found in the *comment* will be removed when the comment is displayed. Case distinctions in the key phrases are ignored.

When this command is issued, the form specified by *form_name* is added to the list of forms. If the form already exists, its description is changed to reflect the new information. Once added, a form is available for use in a print request, except where access to the form has been restricted, as described under the `-u` option. A form may also be allowed to be used on certain printers only.

A description of each form attribute is below:

`Page length and Page Width`
: Before printing the content of a print request needing this form, the generic interface program provided with the LP print service will initialize the physical printer to handle pages $scaled\ -decimal\ -number_1$ long, and $scaled\ -decimal\ -number_2$ wide using the printer type as a key into the `terminfo` database.

  The page length and page width will also be passed, if possible, to each filter used in a request needing this form.

`Number of pages`
: Each time the alignment pattern is printed, the LP print service will attempt to truncate the *content* to a single form by, if possible, passing to each filter the page subset of 1-*integer*.

`Line pitch` and `Character pitch`
: Before printing the content of a print request needing this form, the interface programs provided with the LP print service will initialize the physical printer to handle these pitches, using the printer type as a key into the `terminfo` database. Also, the pitches will be passed, if possible, to each filter used in a request needing this form. $scaled\ -decimal\ -number_3$ is in lines per centimeter if a `c` is appended, and lines per inch otherwise; similarly, $scaled\ -decimal\ -number_4$ is in characters per centimeter if a `c` is appended, and characters per inch otherwise. The character pitch can also be given as `elite` (12 characters per inch), `pica` (10 characters per inch), or `compressed` (as many characters per inch as possible).

`Character set choice`
: When the LP print service alerts an administrator to mount this form, it will also mention that the print wheel *print-wheel* should be used on those printers that take print wheels. If printing with this form is to be done on a printer that has selectable or loadable character sets instead of print wheels, the interface programs provided with the LP print service will automatically select or load the correct character set. If `mandatory` is appended, a user is not allowed to select a different character set for use with the form; otherwise, the character set or print wheel named is a suggestion and a default only.

`Ribbon color`
: When the LP print service alerts an administrator to mount this form, it will also mention that the color of the ribbon should be *ribbon-color*.

`Comment`
: The LP print service will display the *comment* unaltered when a user asks about this form [see `lpstat(1)`].

`Alignment pattern`
: When mounting this form an administrator can ask for the *content* to be printed repeatedly, as an aid in correctly positioning the preprinted form. The optional *content-type* defines the type of printer for which

content had been generated. If *content-type* is not given, `simple` is assumed. Note that the *content* is stored as given, and will be readable only by the user `lp`.

When an existing form is changed, items missing in the new information are left as they were. When a new form is added with this command, missing items will get the following defaults:

Page Length: `66`
Page Width: `80`
Number of Pages: `1`
Line Pitch: `6`
Character Pitch: `10`
Character Set Choice: `any`
Ribbon Color: `any`

Deleting a Form

The `–x` option is used to delete the form *form_name* from the LP print service.

Listing Form Attributes

The `–l` option is used to list the attributes of the existing form *form_name*. The attributes listed are those described under **Adding or Changing a Form**. Because of the potentially sensitive nature of the alignment pattern, only the administrator can examine the form with this command. Other people may use the `lpstat` command to examine the non-sensitive part of the form description.

Allowing and Denying Access to a Form

The `–u` option, followed by `allow:`*login-ID-list* or `–u deny:`*login-ID-list* lets you determine which users will be allowed to specify a particular form with a print request. This option can be used with the `–F` or "–" option, each of which is described above under **Adding or Changing a Form**.

The *login-ID-list* argument may include any or all of the following constructs:

login-ID
: A user on any system

system_name`!`*login-ID*
: A user on system *system_name*

system_name`!all`
: All users on system *system_name*

`all!`*login-ID*
: A user on all systems

`all` All users on all systems

The LP print service keeps two lists of users for each form: an "allow-list" of people allowed to use the form, and a "deny-list" of people that may not use the form. With the `–u allow` option, the users listed are added to the allow-list and removed from the deny-list. With the `–u deny` option, the users listed are added to the deny-list and removed from the allow-list. (Both forms of the `–u` option can be run together with the `–F` or the `–` option.)

If the allow-list is not empty, only the users in the list are allowed access to the form, regardless of the contents of the deny-list. If the allow-list is empty but the deny-list is not, the users in the deny-list may not use the form, (but all others may use it). All users can be denied access to a form by specifying `–u deny:all`. All users can be allowed access to a form by specifying `–u allow:all`. (This is the default.)

Setting an Alert to Mount a Form

The `–f` *form_name* option is used with the `–A` *alert_type* option to define an alert to mount the form when there are queued jobs which need it. If this option is not used to arrange alerting for a form, no alert will be sent for that form.

The method by which the alert is sent depends on the value of the *alert_type* argument specified with the `–A` option. The *alert_types* are:

`list` Displays the type of the alert for the form on standard output. No change is made to the alert.

`mail` Sends the alert message via the `mail` command to the administrator.

`write`
Writes the message, via the `write` command, to the terminal on which the administrator is logged in. If the administrator is logged in on several terminals, one is arbitrarily chosen.

`quiet`
Messages are not sent for the current condition. An administrator can use this option to temporarily stop receiving further messages about a known problem. Once the form *form_name* has been mounted and subsequently unmounted, messages will again be sent when the number of print requests reaches the threshold specified by the `-Q` option.

`none` Messages are not sent until the `-A` option is given again with a different *alert_type* (other than `quiet`).

shell-command
Runs the *shell-command* each time the alert needs to be sent. The shell command should expect the message in standard input. If there are blanks embedded in the command, enclose the command in quotes. Note that the `mail` and `write` values for this option are equivalent to the values `mail` *login-ID* and `write` *login-ID* respectively, where *login-ID* is the current administrator's name. This is the login name of the person submitting this command unless the `su` command was used to change to another login-ID. If the `su` command has been used to change the user ID, the *user-name* for the new ID is used.

The message sent appears as follows:

`The form` *form_name* `needs to be mounted on the printer(s):` *printer* `(`$integer_1$ *requests*`).` $integer_2$
`print requests await this form.`
`Use the` *ribbon-color* `ribbon.`
`Use the` *print-wheel* `print wheel, if appropriate.`

The printers listed are those that the administrator had earlier specified were candidates for this form. The number $integer_1$ listed next to each printer is the number of requests eligible for the printer. The number $integer_2$ shown after the list of printers is the total number of requests awaiting the form. It will be less than the sum of the other numbers if some requests can be handled by more than one printer. The *ribbon-color* and *print-wheel* are those specified in the form description. The last line in the message is always sent, even if none of the printers listed use print wheels, because the administrator may choose to mount the form on a printer that does use a print wheel.

Where any color ribbon or any print wheel can be used, the statements above will read:

```
Use any ribbon.
Use any print-wheel.
```

If *form_name* is `any`, the alerting defined in this command applies to any form for which an alert has not yet been defined. If *form_name* is `all`, the alerting defined in this command applies to all forms.

If the `-W` option is not given, the default procedure is that only one message will be sent per need to mount the form. Not specifying the `-W` option is equivalent to specifying `-W once` or `-W 0`. If *minutes* is a number greater than 0, an alert will be sent at intervals specified by *minutes*.

If the `-Q` option is also given, the alert will be sent when a certain number (specified by the argument *requests*) of print requests that need the form are waiting. If the `-Q` option is not given, or the value of *requests* is `1` or `any` (which are both the default), a message is sent as soon as anyone submits a print request for the form when it is not mounted.

Listing the Current Alert

The `-f` option, followed by the `-A` option and the argument `list` is used to list the type of alert that has been defined for the specified form *form_name*. No change is made to the alert. If *form_name* is recognized by the LP print service, one of the following lines is sent to the standard output, depending on the type of alert for the

form:

- `When` *requests* requests are queued: □
  `alert with` *shell-command* `every` *minutes* `minutes`
- `When` *requests* requests are queued: □
  `write to` *user-name* `every` *minutes* `minutes`
- `When` *requests* `requests are queued:` □
  `mail to` *user-name* `every` *minutes* `minutes`
- `No alert`

The phrase `every` *minutes* `minutes` is replaced with `once` if *minutes* (`–W` *minutes*) is 0.

Terminating an Active Alert

The `–A quiet` option is used to stop messages for the current condition. An administrator can use this option to temporarily stop receiving further messages about a known problem. Once the form has been mounted and then unmounted, messages will again be sent when the number of print requests reaches the threshold *requests*.

Removing an Alert Definition

No messages will be sent after the `–A none` option is used until the `–A` option is given again with a different *alert_type*. This can be used to permanently stop further messages from being sent as any existing alert definition for the form will be removed.

SEE ALSO

`lp`(1), `lpadmin`(1M), `terminfo`(4).

System Administrator's Reference Manual

lpsched(1M)

NAME

`lpsched, lpshut, lpmove -` start/stop the LPprint service and move requests

SYNOPSIS

`/usr/lib/lp/lpsched`

`lpshut`

`lpmove` *requests dest*

`lpmove` *dest1 dest2*

DESCRIPTION

`lpsched` starts the LP print service; this can be done only by `root` or `lp`.

`lpshut` shuts down the print service. All printers that are printing at the time `lpshut` is invoked will stop printing. When `lpsched` is started again, requests that were printing at the time a printer was shut down will be reprinted from the beginning.

`lpmove` moves requests that were queued by `lp` between LP destinations. The first form of the `lpmove` command shown above (under **SYNOPSIS**) moves the named *requests* to the LP destination *dest*. *requests* are request-IDs as returned by `lp`. The second form of the `lpmove` command will attempt to move all requests for destination dest1 to destination dest2; `lp` will then reject any new requests for dest1.

Note that when moving requests, `lpmove` never checks the acceptance status [see `accept`(1M)] of the new destination. Also, the request-IDs of the moved request are not changed, so that users can still find their requests. The `lpmove` command will not move requests that have options (content type, form required, and so on) that cannot be handled by the new destination.

If a request was originally queued for a class or the special destination `any`, and the first form of `lpmove` was used, the destination of the request will be changed to *new-destination*. A request thus affected will be printable only on *new-destination* and not on other members of the `class` or other acceptable printers if the original destination was `any`.

FILES

`/var/spool/lp/*`

SEE ALSO

`enable`(1), `lp`(1), `lpstat`(1), `accept`(1M), `lpadmin`(1M).

System Administrator's Reference Manual

lpsystem(1M)

NAME

`lpsystem -` register remote systems with the print service

SYNOPSIS

`lpsystem` [`–t` *type*] [`–T` *timeout*] [`–R` *retry*] [`–y` "*comment*"] *system_name* [*system_name ...*]

`lpsystem –l` [*system_name ...*]

`lpsystem –r` *system_name* [*system_name ...*]

`lpsystem –A`

DESCRIPTION

The `lpsystem` command is used to define parameters for the LP print service, with respect to communication (via a high-speed network such as STARLAN or TCP/IP) with remote systems. Only a privileged user (that is, the owner of the login `root`) may execute the `lpsystem` command.

Specifically, the `lpsystem` command is used to define remote systems with which the local LP print service can exchange print requests. These remote systems are described to the local LP print service in terms of several parameters that control communication: type, retry and timeout. These parameters are defined in `/etc/lp/Systems`. You can edit this file with a text editor (such as `vi`) but editing is not recommended.

The *type* parameter defines the remote system as one of two types: `s5` or `bsd`. The default type is `s5`.

The *timeout* parameter specifies the length of time (in minutes) that the print service should allow a network connection to be idle. If the connection to the remote system is idle (that is, there is no network traffic) for *N* minutes, then drop the connection. (When there is more work the connection will be re-established.) Legal values are `n`, `0`, and *N*, where *N* is an integer greater than 0. The value `n` means "never time out"; `0` means "as soon as the connection is idle, drop it". The default is `n`.

The *retry* parameter specifies the length of time to wait before trying to re-establish a connection to the remote system, when the connection was dropped abnormally (that is, a network error). Legal values are `n`, `0`, and *N*, where *N* is an integer greater than 0 and it means "wait *N* minutes before trying to reconnect". (The default is 10 minutes.) The value `n` means "do not retry dropped connections until there is more work"; `0` means "try to reconnect immediately".

The *comment* argument allows you to associate a free form comment with the system entry. This is visible when `lpsystem –l` is used.

system-name is the name of the remote system from which you want to be able to receive jobs, and to which you want to be able to send jobs.

The command `lpsystem –l` [*system_name*] will print out a description of the parameters associated with *system_name* (if a system has been specified), or with all the systems in its database (if *system_name* has not been specified).

The command `lpsystem –r` *system_name* will remove the entry associated with *system_name*. The print service will no longer accept jobs from that system or send jobs to it, even if the remote printer is still defined on the local system.

The command `lpsystem –A` will print out the TCP/IP address of the local machine in a format to be used when configuring the local port monitor to accept requests from a BSD system.

NOTES

With respect to `/etc/lp/Systems`, this information is relatively minimal with respect to controlling network communications. Network addresses and services are handled by the `Netconfig` and `Netdir` facilities. Port monitors handle listening for remote service requests and routing the connection to the print service.

If the `Netconfig` and `Netdir` facilities are not set up properly, out-bound remote print service probably will not work. Similarly, if the local port monitors are not set up to route remote print requests to the print service,

then service for remote systems will not be provided.

With respect to the semantics of the *timeout* and *retry* values, the print service uses one process for each remote system with which it communicates, and it communicates with a remote system only when there is work to be done on that system or work being sent from that system.

The system initiating the connection is the "master" process and the system accepting the connection is the "slave" process. This designation serves only to determine which process dies (the slave) when a connection is dropped. This helps prevent there from being more than one process communicating with a remote system. Furthermore, all connections are bi-directional, regardless of the master/slave designation. You cannot control a system's master/slave designation. Now, keeping all this information in mind, if a master process times out, then both the slave and master will exit. If a slave times out, then it is possible that the master may still live and retry the connection after the retry interval. Therefore, one system's resource management strategy can effect another system's strategy.

With respect to `lpsystem –A`: a BSD system (described with `–t bsd`) can be connected to your system only via TCP/IP, and print requests from a BSD system can come in to your machine only via a special port (515). The address given to you from `lpsystem` will be the address of your system and port 515. This address is used by your TCP/IP port monitor [see `sacadm`(1M) and `nlsadmin`(1M)] to "listen" on that address and port, and to route connections to the print service. The important point here is that this is where you get the address referred to in that procedure.

The command `lpsystem –A` will not work if your system name and IP address are not listed in `/etc/inet/hosts` and the printer service is not listed in `/etc/inet/services`.

FILES

`/var/spool/lp/*`
`/etc/lp/*`

SEE ALSO

`netconfig`(4) .

System Administrator's Reference Manual

lpusers(1M)

NAME

`lpusers -` set print queue priorities

SYNOPSIS

`lpusers –d` *priority_level*

`lpusers –q` *priority_level* `–u` *login_ID_list*

`lpusers –u` *login_ID_list*

`lpusers –q` *priority_level*

`lpusers –l`

DESCRIPTION

The `lpusers` command is used to set limits to the queue priority level that can be assigned to jobs submitted by users of the LP print service.

The first form of the command (with `–d`) sets the priority default for the system to *priority_level*, where *priority_level* is a value between 0 to 39, with 0 being the highest priority. If a user does not specify a priority level with a print request [see `lp(1)`], the default priority is used. Initially, the default priority level is 20.

The second form of the command (with `–q` and `–u`) sets the default highest *priority_level* (0-39) that the users in *login_ID_list* can request when submitting a print request. The *login_ID_list* argument may include any or all of the following constructs:

login-ID
: A user on any system

system_name`!`*login-ID*
: A user on the system *system_name*

system_name`!all`
: All users on system *system_name*

`all!`*login-ID*
: A user on all systems

`all`
: All users on all systems

Users that have been given a limit cannot submit a print request with a higher priority level than the one assigned, nor can they change a request already submitted to have a higher priority. Any print requests submitted with priority levels higher than allowed will be given the highest priority allowed.

The third form of the command (with `–u`) removes any explicit priority level for the specified users.

The fourth form of the command (with `–q`) sets the default highest priority level for all users not explicitly covered by the use of the second form of this command.

The last form of the command (with `–l`) lists the default priority level and the priority limits assigned to users.

SEE ALSO

`lp(1)` .

macref(1M) - mvdir(1M)

System Administrator's Reference Manual

macref(1M)

NAME

`macref` - produce cross-reference listing of macro files

SYNOPSIS

`macref [ –t ] [ –s ] [ –n ] [ –– ]` *file* ...

DESCRIPTION

`macref` reads the named files (which are assumed to be `nroff`(1M) or `troff`(1M) input), and produces a cross-reference listing of the symbols in the input.

A `–t` on the command line causes a macro table of contents to be printed. The `–s` option causes symbol-use statistics to be printed. The `–n` option causes one line to be printed for each reference to a symbol. The options may be grouped behind one –. You may use "––" to delimit the end of options. `macref` does not accept – as standard input.

The default output is a list of the symbols found in the input, each accompanied by a list of all references to that symbol. `macref` lists the symbols alphabetically in the leftmost column, with the references following to the right. Each reference is given in the form:

[[(*NMname*)] *Mname*–] *type lnum* [#]

where the fields have the following meanings:

Mname
: The name of the macro within which the reference occurs. This field is missing if the reference occurs outside a macro. Any names listed in the *NMname* part are macros within which *Mname* is defined.

type
: The type associated, by context, with this occurrence of the symbol. The types may be:

  `r` request

  `m` macro

  `d` diversion

  `s` string

  `n` number register

  `p` parameter (e.g. \$*x* is a parameter reference to *x*. Note that parameters are never modified, and that the only valid parameter symbol names are 1, 2, ... 9).

lnum
: The line number on which the reference occurred.

#
: This reference modifies the value of the symbol.

Generated names are listed under the artificial symbol name "~sym".

SEE ALSO

`nroff`(1M) , `troff`(1M) , `man`(5) , `mm`(5) , `mv`(5) .

System Administrator's Reference Manual

mail.local(1M)

NAME

`mail.local` - store mail in a mailbox

SYNOPSIS

`mail.local` [–f *from*] *user* ...

DESCRIPTION

`mail.local` reads the standard input up to an end-of-file and appends it to each user's mail file. The user must be a valid user name.

OPTIONS

–f *from*
: Specify the sender's name.

Individual mail messages in the mailbox are delimited by an empty line followed by a line beginning with the string `From`. A line containing the string `From`, the sender's name and a time stamp is prepended to each delivered mail message. A blank line is appended to each message. A greater-than character (`>`) is prepended to any line in the message which could be mistaken for a `From` delimiter line.

The mail files are exclusively locked while mail is appended.

If the "biff" service is returned by `getservbyname`(3N) , the "biff" server is notified of delivered mail.

DIAGNOSTICS

The `mail.local` utility exits `0` on success, and `>0` if an error occurs.

ENVIRONMENT VARIABLES

`TZ` Used to set the appropriate time zone on the timestamp.

FILES

`/tmp/local.`*XXXXXX*
: temporary files

`/var/mail/`*user*
: user's mailbox directory

SEE ALSO

`mail`(1-ucb) , `comsat`(1M) , `sendmail`(1M) , `getservbyname`(3N) .

System Administrator's Reference Manual

mail_pipe(1M)

NAME

`mail_pipe` - invoke recipient command for incoming mail

SYNOPSIS

`mail_pipe` [`-x` *debug_level*] `-r` *recipient* `-R` *path_to_sender* `-c` *content_type* `-S` *subject*

DESCRIPTION

When a new mail message arrives, the `mail` command first checks if the recipient's mailbox indicates that the message is to be forwarded elsewhere (to some other recipient or as the input to some command). If the message is to be piped into a recipient-specified command, `mail` invokes `mail_pipe` to do some validation and then execute the command in the context of the recipient.

OPTIONS

`-x` *debug_level*
: Turn on debugging for this invocation. See the description of the `-x` option for the `mail` command for details.

`-r` *recipient*
: The recipient's login ID.

`-R` *path_to_sender*
: The return address to the message's originator.

`-c` *content_type*
: The value of the `Content-Type:` header line in the message.

`-S` *subject*
: The value of the `Subject:` header line in the message if present.

`mail_pipe` is installed as a setuid-to-root program, thus enabling itself to change its user and group ids to that of the recipient as necessary.

When invoked, `mail_pipe` performs the following steps (if a step fails, the exit code is noted as [*N*]):

- Validate invocation arguments [1].
- Verify that recipient name is ≤ 14 characters long [2].
- Verify that the setgid flag for the recipient mailbox is set [3].
- Open `/var/mail/`*recipient* [4].
- Verify that recipient's mailbox starts with the string `Forward to` [5].
- Find pipe symbol indicating start of command string in recipient mailbox [6].
- Find entry for recipient in `/etc/passwd` [7].
- Set gid to recipient's gid [8].
- Set uid to recipient's uid [9].
- Change current directory to recipient's login directory [10].
- Allocate space to hold newly `exec`'ed environment for recipient command [11].
- Parse the recipient command, performing any `%`*keyword* expansions required. See the "Forwarding mail" section of `mail(1)`, for more information regarding `%`*keyword* substitutions [12].
- Execute recipient command (13 if `exec` fails, otherwise exit code from recipient command itself).

FILES

`/etc/passwd`
: to identify sender and locate recipients

`/var/mail/`*recipient*
: incoming mail for *recipient*; that is, the mail file

`/tmp/MLDBG*`
: debug trace file

`/usr/lib/mail/mail_pipe`
: `mail_pipe` program

SEE ALSO

`mail`(1), `notify`(1), `vacation`(1).

System Administrator's Reference Manual

mailq(1M)

NAME

`mailq -` print the mail queue

SYNOPSIS

`mailq [-v]`

DESCRIPTION

`mailq` prints a summary of the mail messages queued for future delivery.

The first line printed for each message shows the internal identifier used on this host for the message, the size of the message in bytes, the date and time the message was accepted into the queue, and the envelope sender of the message. The second line shows the error message that caused this message to be retained in the queue; it will not be present if the message is being processed for the first time. The following lines show message recipients, one per line.

`mailq` is identical to `sendmail -bp`.

OPTIONS

`-v` Prints verbose information. This adds the priority of the message and a single character indicator ("+" or blank) indicating whether a warning message has been sent on the first line of the message. Additionally, extra lines may be intermixed with the recipients indicating the "controlling user" information; this shows who will own any programs that are executed on behalf of this message and the name of the alias this command expanded from, if any.

DIAGNOSTICS

The `mailq` utility exits `0` on success, and `>0` if an error occurs.

SEE ALSO

`sendmail`(1M) .

System Administrator's Reference Manual

mailstats(1M)

NAME

`mailstats` - display mail statistics

SYNOPSIS

`mailstats [–o] [–C` *cffile* `] [–f` *stfile* `]`

DESCRIPTION

The `mailstats` program is supplied with `sendmail(1M)` to provide an easy way to view the contents of the `sendmail.st` file.

The output of `mailstats` looks like this:

```
Statistics from Thu Sep  7 13:52:50 1995
 M msgsfr bytes_from  msgsto   bytes_to  Mailer
 1      0          0K      4          8K  *file*
 3    324       2466K   1297       4073K  local
 4   1082       4425K     37       1153K  relay
 5      1          3K    103       1133K  smtp
=======================================
 T   1407       6894K   1441       6367K
```

In the first line, the time at which statistics started being kept is displayed. Then, the statistics for each mailer are displayed on a single line, each with the following whitespace separated fields:

`M`
: The mailer number.

`msgsfr`
: Number of messages from the mailer.

`bytes_from`
: Kbytes from the mailer.

`msgsto`
: Number of messages to the mailer.

`bytes_to`
: Kbytes to the mailer.

`Mailer`
: The name of the mailer.

After this display, a line totaling the values for all of the mailers is displayed, separated from the previous information by a line containing only equals `=` characters.

OPTIONS

`–C` *cffile*
: Read the specified file instead of the default `sendmail.cf` file.

`–f` *stfile*
: Read the specified statistics file instead of the statistics file specified in the `sendmail.cf` file.

`–o`
: Don't display the name of the mailer in the output.

DIAGNOSTICS

The `mailstats` utility exits `0` on success, and `>0` if an error occurs.

FILES

`/etc/sendmail.cf`
: The default `sendmail` configuration file

`/var/log/sendmail.st`

The default `sendmail` statistics file

SEE ALSO

`mailq`(1M), `sendmail`(1M).

System Administrator's Reference Manual

makedict(1M)

NAME

`makedict` - build DBM dictionary

SYNOPSIS

`makedict` *dbm_dictionary*

DESCRIPTION

`makedict` reads from standard input and generates a DBM database called *dbm_dictionary*, which contains the illegal passwords for the `passwd` command. If *dbm_dictionary* does not yet exist, it is created; otherwise, the entries are added to the dictionary.

The dictionary can be read using the `viewdict`(1M) command.

FILES

dbm_dictionary`.dir`
: DBM data base containing a bit map

dbm_dictionary`.pag`
: DBM data base containing all data

SEE ALSO

`passwd`(1), `viewdict`(1M), `dbm`(3C).

System Administrator's Reference Manual

makemap(1M)

NAME

`makemap` - create database maps for sendmail

SYNOPSIS

`makemap` [`–Nforv`] *maptype mapname*

DESCRIPTION

`makemap` creates the database maps used by the keyed map lookups in `sendmail(1M)`. It reads input from standard input and outputs it to the *mapname* specified.

`makemap` handles up to three different database formats, selected using the *maptype* parameter. They may be

`dbm` DBM format maps.

`btree`
B-Tree format maps.

`hash` Hash format maps.

In all cases, `makemap` reads lines from standard input consisting of two words separated by white space. The first is the database key, the second is the value. The value may contain "`%n`" strings to indicated parameter substitution. Literal percent signs should be doubled ("`%%`"). Blank lines and lines beginning with "#" are ignored.

OPTIONS

`–N` Includes the null byte that terminates strings in the map. This must match the `–N` option in the "K" line in `sendmail.cf`.

`–f` Normally all uppercase letters in the key are folded to lowercase. This option disables that behavior. This is intended to mesh with the `–f` option in the "K" line in `sendmail.cf`. The value is never case folded.

`–o` Appends something to an old file. This allows you to augment an existing file.

`–r` Allows existing keys to be replaced. Normally `makemap` complains if you repeat a key, and will not do the insert.

`–v` Verbosely print what it is doing.

SEE ALSO

`sendmail(1M)`.

System Administrator's Reference Manual

mapchan(1M)

NAME

`mapchan` - display and configure terminal mapping

SYNOPSIS

Format 1

`mapchan` [`–o` | `–d`] [*channel*]

Format 2

`mapchan –s` [`–f` *mapfile*] [*channel ...*]

Format 3

`mapchan –sn` [*channel ...*]

Format 4

`mapchan –sa` [`–f` *mapfile*]

DESCRIPTION

The `mapchan` mechanism supports the mapping of characters input and output on terminals. On the input side, single characters can be mapped to other characters (1-to-1 mapping), and compose key sequences and dead key sequences (see below) can be defined. On the output side, single characters can be mapped to one or more characters (1-to-N mapping).

The `mapchan` mechanism is typically useful for supporting terminals which have no compose or dead key sequences of their own or for converting output characters to device-specific character sequences.

The `mapchan` command is used to display and configure the mapping table of one or more terminals.

Displaying mapping tables (Format 1)

The `mapchan` command in its first form writes the mapping table of terminal *channel* to standard output. If *channel* is not specified, `mapchan` uses the terminal associated with standard input, standard output or standard error.

channel can be given as an absolute path name or as a path name relative to the `/dev` directory (e.g. `/dev/term/tty000` or `term/tty000`).

The mapping table is output in a form which can be used as an input file (*mapfile*) for another `mapchan` call. The map file format is described on the `mapchan(4)` manual page.

By default, non-alphanumeric characters in the mapping table output are shown in hexadecimal. This can be changed with the following options:

`–o` Non-alphanumeric characters are shown in octal. This option must not be combined with any other options.

`–d` Non-alphanumeric characters are shown in decimal. This option must not be combined with any other options.

The /etc/default/mapchan configuration file

The `/etc/default/mapchan` configuration file defines assignments of mapping tables to terminals. The `mapchan` command evaluates these assignments when configuring terminals (see below).

An assignment in the configuration file consists of a single line comprising the name of a terminal and the name of a map file. The name of the terminal must begin in the first column of the line and the two names must be separated by one or more blanks. Blank lines and lines beginning with the # character are ignored.

The `/etc/default/mapchan` file will look something like this:

```
# Default assignment
term/tty013  mapfile1
```

`term/tty300 mapfile2`

The terminal and map file specifications may be either absolute or relative. Relative terminal names are looked for in the `/dev` directory; relatively specified map files must reside in the `/usr/lib/mapchan` directory.

An assignment without a map file specification is equivalent to disabling the `mapchan` mechanism for the specified terminal.

Configuring mapping tables (Formats 2, 3 and 4)

The `–f`, `–n` and `–a` options of the `mapchan` command all have to be combined with the `–s` option. There are various ways of specifying the terminals and the map files.

If no terminals are specified as *channel* ... with the `mapchan –s` command, `mapchan` uses the terminal associated with standard input, standard output or standard error.

If the `–f` *mapfile* option is added to the `mapchan –s` command, the terminals specified as *channel* ... are configured on the basis of the mapping table defined in the given map file. The map file format is described on the `mapchan(4)` manual page.

If `mapchan –s` is issued without the `–f` *mapfile* option, the `mapchan` command uses the map files assigned to the specified terminals in the `/etc/default/mapchan` configuration file.

Using the `–n` option instead of the `–f` *mapfile* option is equivalent to specifying an empty mapping table, thereby disabling the `mapchan` mechanism for the specified terminals.

If the `–s` option is combined with the `–a` option, no terminals must be specified. This form of the command causes all the terminals listed in the `/etc/default/mapchan` configuration file to be configured, using either the map file named with the `–f` option or the map files assigned to the terminals in the configuration file. It is usually best for use of the `–a` option to be restricted to the system administrator.

General

Ordinary users can only configure terminals of which they are the owner; the system administrator is authorized to configure any terminal.

The first time a terminal is opened, the `mapchan` mechanism is disabled. Thus the `mapchan` setting of a terminal is not retained beyond the end of a session.

A dead key sequence is a sequence of two keystrokes mapped to a single character. The first character of a dead key sequence must be identified as a dead key in the mapping table. The characters usually defined as dead keys are the diacritics. Thus the dead key sequence 'e could be mapped to the character é.

A compose key sequence is a sequence of three keystrokes mapped to a single character. A compose key sequence must always start with the compose key defined in the mapping table.

Errors on input may have the following causes:

- If the character combination (dead or compose key sequence) is not defined in the map file, the characters are discarded.
- If a dead key sequence is interrupted by a dead key, both characters are discarded.
- If a dead key sequence is interrupted by the compose key, the dead key is discarded and the compose sequence is evaluated.
- If a compose sequence is interrupted by the compose key, then
  - the first compose sequence is discarded
  - the second compose sequence is evaluated

A compose key sequence may include dead key characters. These are then treated as normal characters.

If the map file includes the keyword "beep", an audible signal will sound in any of the above situations.

NOTES

As the `mapchan` mechanism is implemented in the STREAMS module `ldterm(7)`, this STREAMS module must be pushed in the terminal interface being configured.

If an invalid mapping table is set on a terminal, it will generally no longer be possible to work on that terminal.

Input characters are always subjected to 1-to-1 mapping, even if they are part of a dead or compose key sequence, except in the case of dead keys used within compose key sequences.

The `ISTRIP` flag [see `stty(1)`] should not be set for a terminal to which a mapping table has been applied, as it causes the eighth bit to be stripped.

Note that configuration of the PC keyboard is done with the `mapkey(1M)` utility.

FILES

`/etc/default/mapchan`

`/usr/lib/mapchan/*`

SEE ALSO

`stty(1)`, `mapkey(1M)`, `mapchan(4)`, `ascii(5)`, `meta-ascii(5)`, `keyboard(7)` [on RM400], `ldterm(7)`.

System Administrator's Reference Manual

mapkey(1M)

NAME

mapkey, mapscrn, mapstr - configure monitor screen mapping

SYNOPSIS

mapkey [–doxV] [*datafile*]

mapscrn [–dg] [*datafile*]

mapstr [–dg] [*datafile*]

DESCRIPTION

mapscrn configures the output mapping of the virtual terminal screen on which it is invoked and can thus be applied only to the console. mapkey and mapstr configure the mapping of the keyboard and string keys (for example, function keys) of a device. In the case of the console the device is the virtual terminal (VT), otherwise it is an alphanumeric terminal (e.g. TC20-V100).

mapscrn and mapstr function on a per-device basis. Mapping on one device does not affect any other device. Setting the default for every device can be done using the –g option. The –g option may be used only by the privileged user.

The mapscrn command sets up a one-to-one character mapping on output. This is also known as the "screen mapping" capability. A screen map is an array of 256 bytes (eight-bit characters). If a character is written to a VT on which screen mapping is enabled, the character actually drawn on the screen is going to be the character in the array at the index value given by the original character.

For example, char foo[256] is the screen map. foo[a] is set to the value "p". If the screen map foo is enabled on the VT, every time the character "a" is written to the terminal, "p" will be displayed.

If a file name is given on the argument line, the respective mapping table is configured from the contents of the input file. If no file is given, the default files /usr/lib/keyboard/keys, /usr/lib/keyboard/strings and /usr/lib/console/screens are used. The –d option causes the mapping table to be read from the kernel instead of written and an ASCII version to be displayed on standard output. The format of the output is suitable for use as input files to mapscrn, mapkey, or mapstr.

The sum of the characters in the strings for mapstr (in the /usr/lib/keyboard/strings file) can be a maximum of 512 (including the end-of-string character).

mapkey, when downloading a mapping table, overwrites the default mapping table for all devices (thus affecting all devices using the default mapping table) unless the –V option is specified. In this case, only the device in which mapkey –V was invoked is affected, and the device will revert to using the default mapping table when it is closed or the user logs out.

Normally only the system administrator can use mapkey to overwrite the default mapping table. The –V option allows ordinary users to do so as well.

When mapkey –d displays the mapping table being used, it is the default mapping table unless the –V option is specified. In this case, mapkey displays the mapping table in use on the device in which mapkey –V was invoked.

Non-superusers can run mapkey and mapstr when the –d option is given.

With the –o or –x options, mapkey displays the mapping table in octal or hexadecimal, respectively.

NOTES

There is no way to specify that the map utilities read their configuration tables from standard input.

FILES

/usr/lib/keyboard/*

/usr/lib/console/*

SEE ALSO

`display(7)` [on RM400], `keyboard(7)` [on RM400], `mapkey(7)` .

System Administrator's Reference Manual

meta2fun(1M)

NAME

`meta2fun` - 8-bit character converter

SYNOPSIS

`meta2fun` < *file*

DESCRIPTION

The `meta2fun` filter replaces (almost) all characters in the 8-bit ISO 8859-1 character set [cf. `meta-ascii`(5)] in the specified *file* with `troff` special characters ("funny characters") [cf. `roff-charset`(5)]. This means that these 8-bit characters can be printed in PostScript format.

EXAMPLES

A typical command looks as follows:

`pic` *file* | tbl | eqn | meta2fun | troff -Tpost ...

NOTES

The following 8-bit ISO 8859-1 characters are **not** defined in PostScript and therefore are also **not** replaced by `meta2fun`:

Hex.	Oct.	Dec.	Graphic	Meaning
A0	240	160	ERROR	NBSP, Non Breaking Space
A6	246	166	ERROR	Broken vertical line
B2	262	178	ERROR	Superscript 2
B3	263	179	ERROR	Superscript 3
B9	271	185	ERROR	Superscript 1
D0	320	208	ERROR	Uppercase Icelandic ETH
DD	335	221	ERROR	Uppercase Y acute accent
DE	336	222	ERROR	Uppercase Icelandic Thorn
F0	360	240	ERROR	Lowercase Icelandic ETH
FD	375	253	ERROR	Lowercase y acute accent
FE	376	254	ERROR	Lowercase Icelandic Thorn

SEE ALSO

`meta-ascii`(5) , `roff-charset`(5) .

System Administrator's Reference Manual

mibtool(1M)

NAME

`mibtool` - edit attribute sets for network stations

SYNOPSIS

`/usr/bin/terms/mibtool` *ans-file* [*prefix*] [*dbg*]

DESCRIPTION

Terminal and printer servers are configured using attribute sets. These attribute sets define the terminal and printer server settings in the same way as the `ttydefs` attributes, which describe the settings for devices connected locally. These attributes form a database, which is arranged in a tree structure by the relevant network station. The nodes are numbered for the purposes of addressing an attribute within a tree. The individual numbers used for addressing are separated by ".". This means that each setting can be made or each attribute can be set in accordance with a value range using this addressing scheme. Settings are made using SNMP on a UNIX host, which under certain conditions configures the entire network.

Each network station has a definition file, in which the available attributes are recorded together with their value range and a comment. This file is available in the `/var/sadm/terms/tsvrtypes/`*Type* directory and is called *Type*`.ans`. The ending ".ans" comes from the definition of the attributes in accordance with the ANS.1 standard. This file defines, among others, the names of the attributes, the numbers, and the data types. The names can be modified freely, the corresponding names and data types, however, cannot.

Using the `mibtool` command, a file is generated from the ".ans" file. This generated file (`/var/sadm/terms/tsvrtypes/`*Type*`/`*Type*`.dat`) contains the node number sequence for each attribute name and the name and corresponding value for each enumeration type. This file is a component of the (`/var/sadm/terms/tsvrtypes/`*Type*`/snmpinfo.dat`) file, which is used by the "setany" and "getone" commands.

Only some of the attributes that can be set are offered for configuration. This set is arranged logically according to attribute, given a name, and combined to form attribute sets of a terminal server or printer server in files that are contained in the directory `/var/sadm/terms/tsvrtypes/`*Type* and have the suffix ".def". The following 5-part tuple is created for each attribute:

marker : *attribute_name* : *datatype* : *value* : *comment*

The marker is freely selectable. The attribute name and its data type must be defined using the ".ans" file. The following data types exist:

`-i` Integer

`-a` IpAddress

`-D` DisplayString

The value can be specified as a fixed value or can be labeled as a variable, which is not specified until the `tsvrset` call is issued (e.g. using CONFIG). The structure of an attribute set can be changed, and can be extended by adding attributes to include settings for functionalities which it has not been possible to configure until now (e.g. to include settings for activating "routing"). A maximum value for the size of an attribute set is defined by the SNMP protocol. This value is currently set to approximately 40 attributes.

If the attribute in question is a single attribute, it has the ending ".0" (zero). If, on the other hand, the attribute is a list, (e.g. `'portTable'`), the ending ".i" is used. This ending is then replaced by the desired port number when the `tsvrset`/`tsvrget` command is called.

The "value" field of an attribute can be defined as follows:

1. *fixed_value*
2. `${`*variable*`}`
3. `${`*variable@initialization*`}`

Variables can be initialized with default values. The corresponding assignment can then be removed when the call is executed. All attributes contained in the "config.def" file must not be changed, as they form the interface to CONFIG.

The formation of attribute sets from attributes should be carried out on the basis of their logical relationship. There are therefore attribute sets for

- basic settings
- ageneral port
- ports according to type
- terminals, printers, modems, serials
- specific terminal, printer, or modem types
- interface settings (e.g. baud rate)

Each attribute set consists of a set of attributes with the same marker. The various markers are combined (whereby the various parts are separated by ":") and are specified using their `l` option when the `tsvrset`/`tsvrget` command is called. The division of the attributes into various parts is necessary to ensure that the attribute set remains manageable despite the large number of connectable devices.

ARGUMENTS

.TP *ans-file Is the file containing the relevant ANS.1 definition for this type ofnetstation*.

prefix
Is the number of the MIB tree for the manufacturer.

dbg Activates the output of information in various levels of detail, depending on the argument ("`X`" or "`Y`").

EXAMPLE

```
mibtool annex8 1.3.6.1.4.1.15
```

FILES

`/var/sadm/terms/tsvrtypes/`*Type*`/`*Type*`.ans`
This file should not be modified

`/var/sadm/terms/tsvrtypes/`*Type*`/`*Type*`.dat`
This file should not be modified

`/var/sadm/terms/tsvrtypes/`*Type*`/config.def`
This file should never be modified with respect to CONFIG

`/var/sadm/terms/tsvrtypes/`*Type*`/misc.def`
This file should not be modified

`/var/sadm/terms/tsvrtypes/`*Type*`/add-on.def`
This file can be modified

SEE ALSO

`tsvrget`(1M), `tsvrset`(1M).

System Administrator's Reference Manual

mkfs(1M)

NAME

`mkfs(generic)` - construct a file system

SYNOPSIS

`mkfs` [`-F` *FSType*] [`-V`] [`-m`] [`-f`] [`-o` *specific_options*] *special* [*operands*]

DESCRIPTION

`mkfs` constructs a file system on a *special* file; *special* must be the first argument and must be a special character device file (for example `/dev/ios0/rsdisk000s1`). The file system is created based on the *FSType*, *specific_options* and *operands* specified on the command line. `mkfs` waits 10 seconds before starting to construct the file system. During this time the command can be aborted by entering a delete (DEL).

operands are *FSType*-specific and the *FSType* specific manual page of `mkfs` should be consulted for a detailed description.

OPTIONS

There are three types of option: *generic_options*, *other_options* and *specific_options*.

generic_options
: Indicates options that are supported not only by the generic but also by the *FST*-specific module of the command (if that command exists).

other_options
: Indicates options that are supported only by the generic command.

specific_options
: Indicates suboptions that are supported only by the *FSType*-specific module [see `mkfs(1M-ufs)` or `mkfs(1M-vxfs)`] of the command. They are specified in a comma-separated list and/or as keyword-attribute pairs for interpretation by the *FSType*-specific module of the command. These suboptions are specified via the `-o` option.

The *generic_options* are:

`-F` Specifies the *FSType* to be constructed. The *FSType* should either be specified here or be determinable from `/etc/vfstab` by matching the *special* with an entry in the table.

`-V` Echoes the complete command line, but does not execute the command. The command line is generated by using the options and arguments provided by the user and adding to them information derived from `/etc/vfstab`. This option should be used to verify and validate the command line. When used with the *FSType* option, it passes the `-V` to the *FSType*-specific module for interpretation.

`-m` Returns the command line which was used to create the file system. The file system must already exist. This option provides a means of determining the command used in constructing the file system. It cannot be used with *generic_options*, *specific_options*, or *operands*. It must be invoked by itself.

`-o` Specifies *FSType*-specific options.

The *other_options* are:

`-f` Fast `mkfs`; skip the 10 second delay.

NOTES

This command may not be supported for all *FSTypes*.

FILES

`/etc/vfstab`
: list of default parameters for each file system

SEE ALSO

`newfs`(1M) , `vfstab`(4) .

FSType-specific modules: `mkfs`(1M-ufs) , `mkfs`(1M-vxfs) . Online call e.g.: "man 1M-ufs mkfs".

System Administrator's Reference Manual

mkfs(1M-ufs)

NAME

`mkfs(ufs)` - construct a ufs file system

SYNOPSIS

`mkfs` [`–F ufs`] [*generic_options*] [`–o` *specific_options*] *special size*

DESCRIPTION

NOTE:

Since the `mkfs` command is awkward to use, it is suggested that the `newfs(1M)` command be used to create `ufs` file systems. The `newfs` command determines the `mkfs` parameters by obtaining the information directly from the disk drive specified via *special*. See the `newfs(1M)` manual page for more information.

`mkfs` constructs a file system by writing to the special file *special* unless the `–o N` flag has been specified. *special* must be a character special device file (for example, `/dev/ios0/rsdisk000s1`). The numeric size *size* (maximum ≤ partition size) specifies the number of sectors in the file system. `mkfs` builds a file system with a root directory and a `lost+found` directory [see `fsck(1M)`]. The number of inodes is calculated as a function of the file system size.

If additional arguments are given, then they are identical to the values specified via the `–o` option described below. If no other arguments are given, `mkfs` will use a set of internal defaults to create a file system. This is not recommended, since the internal defaults will undoubtedly be incorrect for the supplied disks.

OPTIONS

`–F ufs`

Specifies the `ufs`-FSType.

generic_options

Options supported by the generic `mkfs` command.

`–o` Specifies the `ufs` specific suboptions. The following suboptions are available:

`N` The file system is not written to the *special* file. This suboption gives all the information as if the file system was created, but it does not create it.

`nsect=`*n*

The number of sectors per track on the disk. The default is `18`.

`ntrack=`*n*

The number of tracks per cylinder on the disk. The default is `9`.

`bsize=`*n*

The primary block size for files on the file system. It must be a power of two, currently selected from `4096`, `8192` or `16384`. The default is `16384`.

`fragsize=`*n*

The fragment size for files on the file system. The `fragsize` represents the smallest amount of disk space that will be allocated to a file. It must be a power of two currently selected from the range `512` to `16384`. The default is `2048`.

It must be possible to generate a block number for every fragment. The maximum block number is `2147483647`. A maximum of 8 fragments (in `fragsize`) can be created for each block (in `bsize`).

`cgsize=`*n*

The number of disk cylinders per cylinder group. This number must be in the range `1` to `32`. The default is `16`. Further requirements result due to the concrete physical disk.

`minfree=`*n*

The minimum percentage of free disk space allowed. Once the file system capacity reaches this threshold, only a privileged user is allowed to allocate disk blocks. The default value is `10%`.

NOTE:

A cylinder group must contain an exact integer number of file system blocks, which is directly related to the rotational layout of the physical disk.

`rps=`*n*

The rotational speed of the disk, in revolutions per second. The default is `60`.

`nbpi=`*n*

The number of bytes for which one inode block is allocated. This parameter is currently set at one inode block for every 2048 bytes.

NOTE:

If you increase the number of inodes in the file system, you may also need to adjust the `cgsize` and/or `bsize` parameters since `mkfs` only administers 2048 inodes per cylinder group. If this is the case, `mkfs` outputs an appropriate message.

`opt=`*c*

Space or time optimization preference; `s` specifies optimization for space, `t` specifies optimization for time. The default is `t`.

`apc=`*n*

The number of alternates per cylinder. The default is `0`.

`gap=`*n*

The expected time (in milliseconds) to service a transfer completion interrupt and initiate a new transfer on the same disk. It is used to decide how much rotational spacing to place between successive blocks in a file. (Also known as rotational delay.) By default, `mkfs` calculates the best delay based on the disk type. If the disk type cannot be determined, then the absolute default is `8`.

NOTES

The value of the `nbpi` operand in the output of `mkfs –m` is always 2048, even if the file system was created with some other value.

Some disks may not support a value for `nsect` less than 2048 bytes.

Some disks may not support the `apc` parameter.

SEE ALSO

`fsck`(1M) , generic `mkfs`(1M) , `newfs`(1M) , `dir`(4-ufs) , `fs`(4-ufs) .

System Administrator's Reference Manual

mkfs(1M-vxfs)
VxFS

NAME

`mkfs(vxfs)` - construct a Veritas File System (VxFS)

SYNOPSIS

`mkfs` [`–F vxfs`] [*generic_options*] [`–o` *specific_options*] *special size*

DESCRIPTION

`mkfs` constructs a file system by writing to the special file *special* unless the `–o N` flag has been specified. The numeric *size* specifies the number of sectors in the file system. If *size* is given as – (a dash), then the entire device size will be used. `mkfs` builds a file system with a root directory and a `lost+found` directory [see `fsck`(1M)]. The number of inodes is calculated as a function of the file system size. No boot program is initialized by `mkfs`.

generic_options indicate options that are supported by the generic `mkfs`(1M) command. *specific_options* indicate suboptions that are supported only by the Veritas-specific module of the `mkfs` command. They are specified in a comma-separated list and/or as keyword-attribute pairs. These suboptions are specified via the `–o` option.

OPTIONS

`–F vxfs`
: Specifies the Veritas file system type (*FSType*).

generic_options
: Options supported by the generic `mkfs` command.

`–o`
: Specifies the Veritas file system specific options in a comma-separated list. The following options are available:

  `N`
  : The file system is not written to the *special* file. This option gives all the information needed to create a file system but does not create it.

  `ninode=`*n*
  : *n* is the number of inodes on the file system, rounded down. The default is the total number of blocks in allocation units divided by four.

  `nau=`*n*
  : *n* is the number of allocation units on the file system. The default is the number of available blocks, after the super block and log area have been allocated, divided by 32768. Each allocation unit will have an equal number of blocks of size `bsize`. This may result in the size of the resulting file system being smaller than that specified by *size*.

  `bsize=`*n*
  : *n* is the block size for files on the file system and represents the smallest amount of disk space that will be allocated to a file. *n* must be a power of two currently selected from the range 1024 to 8192. The default is 1024, or the sector size, whichever is larger. In the case of file systems >32 GB, the block size depends on the maximum file size (see `maxfilesize` option) as well as on the size of the device file (it must be possible to display the block numbers in a "signed 32-bit-long").

  `logsize=`*n*
  : *n* is the number of blocks to allocate for an activity logging area. *n* must be in the range of 32 to 32768. The default is 256.

  `maxfilesize=`*n*
  : *n* is the maximum size of a file in the file system to be created (unit: gigabyte).

n must be a power of two in the range 32 to 16384. By default, `maxfilesize` is at least as large as the file system.

Note: If the maximum file size is less than the size of the file system, it is not possible for a file to be larger than the `maxfilesize`, even if there is enough disk space available (see the notes at the end of this manual page).

`iaddrlen=`*n*

n is the size of the indirect address extents for setting up the indirect levels (see "VxFS System Administrator's Guide") (unit: blocks). *n* must be a power of two in the range 1 to 64.

`iaddrlen * bsize` must be ≥ 8 KB and ≤ 64 KB (see also the notes at the end of this manual page).

`indsize=`*n*

n is the size of the indirect extents in the indirect levels (see "VxFS System Administrator's Guide") (unit: blocks). *n* must be a power of two in the range 1 to 1024.

`indsize * bsize` must be ≥ 8 KB (see also the notes at the end of this manual page).

`ausize=`*n*

n is the size in `bsize` blocks of size of an allocation unit. This is an alternate way of specifying the number of allocations units. This option may not be used in conjunction with the `nau` option. With this option, the last allocation unit on the file system may be shorter than the others. If the last allocation unit on the file system is not long enough to contain an entire allocation unit header, the resulting size of the file system will be to the end of the last complete allocation unit.

`aufirst=`*n*

n is the starting block number in `bsize` blocks of size of the first allocation unit. This option allows the allocation units to be aligned to a particular boundary, such as a cylinder boundary.

`aupad=`*n*

The size in `bsize` blocks of size to leave between the end of the inode list and the first data block in each allocation unit. This option allows the data blocks of an allocation unit to be aligned to a particular boundary, such as a cylinder boundary.

NOTES

The following tables show the relationship between `bsize`, `iaddrlen` and `indsize` with regard to `maxfilesize`. If `iaddrlen` and `indsize` are not specified, `mkfs` uses the following strategy to represent the maximum file size:

`iaddrlen` is increased first: If it is not sufficient for `maxfilesize`, `indsize` is then increased and `iaddrlen` is set to the lowest possible value. This procedure reduces the space requirement for managing the data in the indirect levels.

The maximum file sizes for different `iaddrlen` values and a `bsize` value of 1 KB:

`bsize` 1 KB				in
`iaddrlen` (blocks) =	8	16	32	64
8	32GB	64GB	128GB	256GB
16	128GB	256GB	512GB	1TB
32	512GB	1TB	2TB	2TB
64	2TB	2TB	2TB	2TB

Maximum file sizes for different `iaddrlen` values and a `bsize` value of 2 KB:

bsize 2 KB				in
iaddrlen	4	8	16	32
(blocks)				
=				
4	32GB	64GB	128GB	256GB
8	128GB	256GB	512GB	1TB
16	512GB	1TB	2TB	4TB
32	2TB	4TB	4TB	4TB

Maximum file sizes for different `iaddrlen` values and a `bsize` value of 4 KB:

bsize 4 KB				in
iaddrlen	2	4	8	16
(blocks)				
=				
2	32GB	64GB	128GB	256GB
4	128GB	256GB	512GB	1TB
8	512GB	1TB	2TB	4TB
16	2TB	4TB	8TB	8TB

Maximum file sizes for different `iaddrlen` values and a `bsize` value of 8 KB:

bsize 8 KB				in
iaddrlen	1	2	4	8
(blocks)				
=				
1	32GB	64GB	128GB	256GB
2	128GB	256GB	512GB	1TB
4	512GB	1TB	2TB	4TB
8	2TB	4TB	8TB	16TB

SEE ALSO

`fsck`(1M), `mkfs`(1M), `fs`(4-vxfs).

System Administrator's Reference Manual

mknod(1M)

NAME

mknod - make a special file

SYNOPSIS

/sbin/mknod *name* b | c *major minor*

/sbin/mknod *name* p

DESCRIPTION

mknod makes a directory entry for a special file.

In the first case, *name* is the special file to be created. The second argument is b to indicate a block-type special file or c to indicate a character-type. The last two arguments are numbers specifying the *major* and *minor* device numbers; these may be either decimal or octal. The assignment of major device numbers is specific to each system. You must have superuser rights to use this form of the command.

The second case is used to create a FIFO (named pipe).

NOTES

If mknod is used to create a device in a remote directory (Remote File Sharing), the major and minor device numbers are interpreted by the server.

LOCALE

The LC_MESSAGES environment variable governs the language in which message texts are displayed.

If LC_MESSAGES is undefined or is defined as the null string, it defaults to the value of LANG. If LANG is likewise undefined or null, the system acts as if it were not internationalized.

If any of the locale variables has an invalid value, the system acts as if none of the variables was set.

The LC_ALL environment variable governs the entire locale. LC_ALL takes precedence over all the other environment variables that affect internationalization.

SEE ALSO

mknod(2) .

System Administrator's Reference Manual

mm(1M)

NAME

mm - print documents formatted with the mm macros

SYNOPSIS

mm [*options*] *files*

DESCRIPTION

Use mm to format documents using nroff and the mm(5) text formatting macro package. mm has options to specify preprocessing by tbl(1M) and/or neqn(1M) , and postprocessing by various terminal mode output filters. The proper pipelines and the required arguments and flags for nroff and mm are generated, depending on the options that you select.

options for mm are given below. Any other arguments or flags (e.g. -rC3) are passed to nroff as appropriate. You may use such options in any order, but you must put them before the *files* arguments. If you do not specify arguments, mm prints a list of its options.

-T*tty_type*
: specifies the type of output terminal. Here is a list of recognized values for *tty_type*:

  450 prepares output for a DASI 450 (default for mm; also equivalent to -T1620.

  450-12
  : prepares output for a DASI 450 in 12-pitch mode.

  300 prepares output for a DASI 300 terminal.

  300-12
  : prepares output for a DASI 300 in 12-pitch mode.

  300s prepares output for a DASI 300S.

  300s-12
  : prepares output for a DASI 300S in 12-pitch mode.

  4014 prepares output for a TEKTRONIX 4014.

  37 prepares output for a TELETYPE+ Model 37 (default for nroff).

  382 prepares output for a DTC-382.

  4000a
  : prepares output for a TRENDATA 4000A.

  X prepares output for an EBCDIC line printer with a TX print train.

  hp prepares output for a Hewlett-Packard HP262x or HP264x (implies -c); also equivalent to -T2621, -T2640, and -T2645.

  43 prepares output for a TELETYPE Model 43 (implies -c).

  40/4 prepares output for a TELETYPE Model 40/4 (implies -c) also equivalent to -T40/4.

  745 prepares output for a Texas Instrument 700 series terminal (implies -c); also equivalent to -T735.

  2631 prepares output for a HP2631 printer (implies -c).

  2631-e
  : same as -T2631, but in expanded mode.

  2631-c
  : same as -T2631, but in compressed mode.

  832 prepares output for an Anderson Jacobson 832 printer (implies -c).

  8510 prepares output for a C.ITOH printer (implies -c).

`tn300`
prepares output for a Terminet 300 printer (implies `-c`).

`lp` prepares output for a device with no reverse or partial line motions or other special features (implies `-c`). If you do not use this option, `mm` uses the value of the shell variable `$TERM` from the environment [see `profile(4)` and `environ(5)`] as the value of *tty_type* if `$TERM` is set; otherwise, `mm` uses `450` as the value of *tty_type*. If you specify several terminal types, the last one takes precedence.

Note: To generate output on a Xerox page printer, use the `mmx` command.

`-12` indicates that the document is to be produced in 12-pitch. You may use this option when `$TERM` is set to one of `300`, `300s`, and `450`. (You must manually set the pitch switch on the DASI 300 and 300s terminals to `12` if you use this option.)

`-c` causes `mm` to invoke `col(1)` ; note that `col(1)` is invoked automatically by `mm` unless *tty_type* is one of `300`, `300s`, `450`, `37`, `4000a`, `382`, `4014`, and `X`.

`-e` causes `mm` to invoke `neqn`; also causes `neqn` to read the `/usr/pub/eqnchar` file [see `eqnchar(5)`].

`-t` causes `mm` to invoke `tbl(1M)` .

`-E` invokes the `-e` option of `nroff`.

As an example, assume that the shell variable `$TERM` is set in the environment to `450`. The two command lines below are then equivalent:

```
mm -t -rC3 -12 file*
tbl file* | nroff -mm -T450-12 -h -rC3
```

`mm` reads the standard input when you specify `-` instead of any file names. (Mentioning other files together with `-` leads to undesired results.) This option allows you to use `mm` as a filter, for example:

`cat` *file* | `mm -`

NOTES

1. `mm` invokes `nroff` with the `-h` flag. With this flag, `nroff` assumes that the terminal has tabs set every 8 character positions.
2. Use the `-o`*list* option of `nroff` to specify ranges of pages to be output. Note, however, that if you invoke `mm` with one or more of the `-e`, `-t`, and `-` options, *together* with the `-o`*list* option of `nroff`, you may cause a harmless "broken pipe" diagnostic if you do not specify the last page of the document in *list*.
3. If you use the `-s` option of `nroff` (to stop between pages of output), use line-feed (rather than return or new-line) to restart the output. The `-s` option of `nroff` does not work with the `-c` option of `mm`, or if `mm` automatically invokes `col(1)` (see `-c` option above).
4. If you lie to `mm` about the kind of terminal its output will be printed on, you will get (often subtle) garbage; however, if you are redirecting output into a file, use the `-T37` option, and then use the appropriate terminal filter when you actually print that file.

DIAGNOSTICS

`mm: no input file`
if none of the arguments is a readable file and you do not use `mm` as a filter.

FILES

`/usr/pub/terminals`
list of supported terminals

SEE ALSO

`mmt(1M)` , `nroff(1M)` , `profile(4)` , `mm(5)` , `nterm(5)` , `environ(5)` .

System Administrator's Reference Manual

mmt(1M)

NAME

mmt, mvt - typeset document with mm/mv macros

SYNOPSIS

mmt [*options*] *files*

mvt [*options*] *files*

DESCRIPTION

The mmt and mvt commands are identical and both format their input using the troff(1M) text formatter. However, the mmt commands uses the mm(5) macro package, while the mvt commands uses the mv(5) macro package. There are options to specify preprocessing by eqn(1M) and/or tbl(1M) and/or pic(1M) and/or grap(1M) . The proper pipelines and the required arguments and flags for troff and for the macro package are generated, depending on the options selected.

OPTIONS

options specific to mmt and mvt are given below. Any other arguments or flags (e.g. –o*list*) are passed to troff. You can put options in any order, but they must appear before any *files*. If you give no arguments or *files*, mmt and mvt will print a list of its options.

–e invokes eqn; also causes eqn to read the /usr/pub/eqnchar file [see eqnchar(5)].

–t invokes tbl.

–p invokes pic.

–g invokes grap, which in turn calls pic.

–T*type*
creates output for troff device *type* [see troff(1M)]. Output for a recognized device (post, X97.ti10p, X97.ti12p, i300, i10, pcl) is sent through the appropriate postprocessor for printing. The default device type is post.

–D*dev*
directs the output to device type *dev*. Supported values for *dev* are:

post PostScript laser printer

lino Linotronic L200P PostScript-based phototypesetter at MH

i10 Imagen Imprint-10 laser printer

i300 Imagen 8/300 laser printer

xrx Xerox 4050/8700/9700 printer

hp Hewlett-Packard LaserJet printer

–z invokes no output filter to process or redirect the output of troff.

When you specify – instead of any *files*, mmt and mvt will read standard input.

NOTE

Use the –o*list* option of troff to specify a subset or range of pages to be output. Note, however, that if you call mmt or mvt with one or more of the –e, –t, –p, –g, and – options *together* with the –o*list* option of troff, you may cause a harmless "broken pipe" diagnostic if the last page of the document is not specified in *list*.

EXAMPLES

To process input with troff and the mm macros, and preview the output on a local PostScript printer:

mmt –Tpost –Dpost *filename*

Then, to print that document on the phototypesetter:

`mmt –Tpost –Dlino` *filename*

To process input with `troff` and the `mv` macros, and generate view graph copies on a local PostScript printer:

`mvt –Tpost –Dpost` *filename*

To process input with `tbl`, `eqn`, `pic`, `grap`, and `troff` with the `mm` macros with a base point size of 10 and send output to the centralized Xerox 4050 printer at your location:

`mmt –g –p –t –e –TX97.ti10p –Dxrx` *filename*

To process input with `tbl` and `troff` using the `mv` macros and generate view graph copies on your Imagen 8/300 printer via a local spooling command:

```
mvt –t –Ti300 –z | spool –dhoney
```

Note: The `–z` option is needed to allow special postprocessing functions or commands.

DIAGNOSTICS

"`mmt: no input file`" or "`mvt: no input file`"
If none of the arguments is a readable file and the command is not used as a filter.

SEE ALSO

`mm`(1M), `troff`(1M), `mm`(5), `mv`(5).

System Administrator's Reference Manual

montbl(1M)

NAME

`montbl` - create monetary database

SYNOPSIS

`montbl` [`-o` *outfile*] *infile*

DESCRIPTION

The `montbl` command takes as input a specification file, *infile*, that describes the formatting conventions for monetary quantities for a specific locale.

OPTIONS

`-o` *outfile*

Write the output on *outfile*; otherwise, write the output on a file named `LC_MONETARY`.

The output of `montbl` is suitable for use by the `localeconv()` function [see `localeconv`(3C)]. Before *outfile* can be used by `localeconv()`, it must be installed in the `/usr/lib/locale/`*locale* directory with the name `LC_MONETARY` by someone who is superuser or a member of group `bin`. *locale* is the locale whose monetary formatting conventions are described in *infile*. This file must be readable by user, group, and other; no other permissions should be set. To use formatting conventions for monetary quantities described in this file, use `setlocale`(3C) to change the locale for category `LC_MONETARY` to *locale* [see `setlocale`(3C)].

Once installed, this file will be used by the `localeconv()` function to initialize the monetary specific fields of a structure of type `struct lconv`. For a description of each field in this structure, see `localeconv`(3C) .

```
struct lconv
{
    char *decimal_point;      /* "." */
    char *thousands_sep;      /* "" (zero length string) */
    char *grouping;           /* "" */
    char *int_curr_symbol;    /* "" */
    char *currency_symbol;    /* "" */
    char *mon_decimal_point;  /* "" */
    char *mon_thousands_sep;  /* "" */
    char *mon_grouping;       /* "" */
    char *positive_sign;      /* "" */
    char *negative_sign;      /* "" */
    char int_frac_digits;     /* CHAR_MAX */
    char frac_digits;         /* CHAR_MAX */
    char p_cs_precedes;       /* CHAR_MAX */
    char p_sep_by_space;      /* CHAR_MAX */
    char n_cs_precedes;       /* CHAR_MAX */
    char n_sep_by_space;      /* CHAR_MAX */
    char p_sign_posn;         /* CHAR_MAX */
    char n_sign_posn;         /* CHAR_MAX */
};
```

The specification file specifies the value of each `struct lconv` member, except for the first three members, *decimal_point*, *thousands_sep*, and *grouping* which are set by the `LC_NUMERIC` category of `setlocale`(3C) . Each member's value is given on a line with the following format:

*keyword*ERROR*value*

where *keyword* is identical to the `struct lconv` field name and *value* is a quoted string for those fields that are a `char *` and an integer for those fields that are an `int`. For example,

```
int_curr_symbol    "ITL."
int_frac_digits    0
```

will set the international currency symbol and the number of fractional digits to be displayed in an internationally formatted monetary quantity to `ITL.` and `0`, respectively.

Blank lines and lines starting with a `#` are taken to be comments and are ignored. A character in a string may be in octal or hex representation. For example, `\141` or `\x61` could be used to represent the letter "a". If there is no specification line for a given structure member, then the default "C" locale value for that member is used (see the values in comments in the `struct lconv` definition above).

Given below is an example of what the specification file for Italy would look like:

```
# Italy
int_curr_symbol     "ITL."
currency_symbol     "L."
mon_decimal_point   ""
mon_thousands_sep   "."
mon_grouping        "\3"
positive_sign       ""
negative_sign       "-"
int_frac_digits     0
frac_digits         0
p_cs_precedes       1
p_sep_by_space      0
n_cs_precedes       1
n_sep_by_space      0
p_sign_posn         1
n_sign_posn         1
```

LOCALE

The `LC_MESSAGES` environment variable governs the language in which message texts are displayed.

If `LC_MESSAGES` is undefined or is defined as the null string, it defaults to the value of `LANG`. If `LANG` is likewise undefined or null, the system acts as if it were not internationalized.

If any of the locale variables has an invalid value, the system acts as if none of the variables was set.

The `LC_ALL` environment variable governs the entire locale. `LC_ALL` takes precedence over all the other environment variables that affect internationalization.

FILES

`/usr/lib/locale/`*`locale`*`/LC_MONETARY`
: `LC_MONETARY` database for *locale*

`/usr/lib/locale/C/montbl_C`
: input file used to construct `LC_MONETARY` in the default locale

SEE ALSO

`localeconv`(3C), `setlocale`(3C).

System Administrator's Reference Manual

mount(1M)

NAME

`mount, umount(generic)` - mount or unmount file systems andremote resources

SYNOPSIS

`mount` [`-F` *FSType*] [`-v` | `-p` | `-a`]

`mount` [`-F` *FSType*] [`-Vr`] [`-o` *specific_options*] {*special* | *mount_point*}

`mount` [`-F` *FSType*] [`-Vr`] [`-o` *specific_options*] *special mount_point*

`umount` [`-F` *FSType*] [`-V`] [`-o` *specific_options*] {*special* | *mount_point*}

`umount` [`-F` *FSType*] `-a`

DESCRIPTION

File systems other than `root ( / )` are considered removable in the sense that they can be either available to users or unavailable. `mount` notifies the system that *special*, a block special device or a remote resource, is available to users from the *mount_point* which must already exist; it becomes the name of the root of the newly mounted *special* or resource.

`mount`, when entered with arguments, validates all arguments except for the device name and invokes a *FSType* specific `mount` module. If invoked with no arguments, `mount` lists all the mounted file systems from the mount table. If, for example, only *special* and/or *mount_point* is specified as an argument, `mount` searches the `/etc/vfstab` file and retrieves the necessary arguments as follows: *FSType*, *special*, *mount_point*, and *specific_options*. It will then invoke the *FSType*-specific `mount` module. The only time `mount` does not search the `/etc/vfstab` file is when all four arguments: *FSType*, *special*, *mount_point*, and *specific_options*, are specified.

Most *FSTypes* do not have a `umount` specific module. If one exists it is executed otherwise the generic module unmounts the file systems. If the `-o` option is specified the `umount` specific module is always executed.

OPTIONS

There are three types of option: *generic_options*, *other_options* and *specific_options*.

generic_options
: Indicate options that are supported not only by the generic but also by the *FST*-specific module of the command (if that command exists).

other_options
: Indicate options that are supported only by the generic command.

specific_options
: Indicate suboptions that are supported only by the *FSType*-specific module [see `mount(1M-ufs)` , `mount(1M-nfs)` or `mount(1M-vxfs)`] of the command. They are specified in a comma-separated list and/or as keyword-attribute pairs for interpretation by the *FSType*-specific module of the command. These suboptions are specified via the `-o` option.

The *generic_options* are:

`-F`
: Specifies the *FSType* on which to operate, or restricts the operation to that *FSType* (the `-pva` options). The *FSType* must be specified or must be determinable from `/etc/vfstab` while mounting a file system.

`-V`
: Echoes the complete command line, but does not execute the command. The command line is generated by using the options and arguments provided by the user and adding to them information derived from `/etc/vfstab`. This option should be used to verify and validate the command line. When used with the *FSType* option, it passes the option to the *FSType*-specific module for interpretation.

`-r`
: Overrides `/etc/vfstab` and mount the file system *read-only*.

–o Specifies *FSType*-specific options.

The *other_options* are:

–v Prints the output in a new style. The new output has the *FSType* and flags displayed in addition to the old output. The *mount_point* and *special* fields are reversed.

–p Prints the list of mounted file systems in the `/etc/vfstab` format.

–a Mounts (unmount) all file systems listed in `/etc/vfstab` (`/etc/mnttab`). If an *FSType* is specified, then (un)mount only file systems of that *FSType*. If mounting, then the `automnt` field in `vfstab` must also be set to `yes` for the mount to occur.

`mount` can be used by any user to list mounted file systems and resources. Only a superuser can mount or unmount file systems.

LOCALE

The `LC_MESSAGES` environment variable governs the language in which message texts are displayed.

If `LC_MESSAGES` is undefined or is defined as the null string, it defaults to the value of `LANG`. If `LANG` is likewise undefined or null, the system acts as if it were not internationalized.

If any of the locale variables has an invalid value, the system acts as if none of the variables was set.

The `LC_ALL` environment variable governs the entire locale. `LC_ALL` takes precedence over all the other environment variables that affect internationalization.

NOTES

The old output format will be phased out in a future release and all output will be in the new –v format. The most significant changes are the addition of two new fields to show the *FSType* and flags and the reversal of the *mount_point* and *special* name.

`mount` adds an entry to the mount table `/etc/mnttab`; `umount` removes an entry from the table.

`mount` refuses to mount a *dirty* file system. Normally, one should run `fsck` to correct this situation. However, a file system may have the ability to be force-mounted as read-only. See the appropriate *FSType*-specific manual page for more information about this.

`mount` refuses to mount two file systems on the same mount point.

FILES

`/etc/mnttab`
: mount table

`/etc/vfstab`
: list of default parameters for each file system

SEE ALSO

`setmnt`(1M), `mnttab`(4), `vfstab`(4).

Manual pages for the *FSType*-specific modules of `mount`.

System Administrator's Reference Manual

mount(1M-devfs)

NAME

`mount(devfs)` - mount `devfs` file systems

SYNOPSIS

`mount` [`–F devfs`] [`–o` *specific_options*] {*special* | *mount_point*}

`mount` [`–F devfs`] [`–o` *specific_options*] *special mount_point*

DESCRIPTION

`mount` attaches a `devfs` file system [cf. `devfs(7)`] to the file system hierarchy at the pathname location *mount_point*, which must already exist. If *mount_point* has any contents prior to the `mount` operation, these remain hidden until the file system is once again unmounted.

OPTIONS

The `–F devfs` option specifies the type of `devfs` file system.

The options listed under *special* are keywords describing how the `devfs` file system should generate the names of the devices under *mount_point*.

The following keywords are defined for *special*:

`/fibre`
: The `devfs` file system creates entries under *mount_point* that comply with the `autoconf` name. The partition Ids "s0" to "s15" are appended to the appropriate `autoconf` names. The letter "r" is generated for raw devices in front of the respective `autoconf` name.

`/serial`
: The `devfs` file system creates entries under *mount_point* that comply with the serial numbers of the connected SCSI devices. The partition Ids "s0" to "s15" are appended to the serial numbers. The letter "r" is generated for raw devices in front of the serial number.

`/vdisk`
: The `devfs` file system creates entries under *mount_point* that comply with the serial numbers of the connected SCSI devices. The character string ".priv" is appended to the serial numbers for the management partition of `VDisk Lite` [cf. `vdisklite(7)`]. The serial number is the user partition managed by `VDisk Lite`. The letter "r" is generated for raw devices in front of the serial number.

`/fcdisk`
: The `devfs` file system creates entries under *mount_point* that comply with the FC (=□Fibre Channel) Loop Id of the connected FC disks. The partition Ids "s0" to "s15" are appended to the FC Loop Id. The letter "r" is generated for raw devices in front of the serial number.

  Attention:

  Disks that have been mounted with `/fcdisk` are no longer supported in WSA (WebSysAdmin).

`–o` *specific_options*
: Specifies `devFS` file system-specific options. If several file system-specific options are specified, they must be separated by a comma.

  Note:

  The `devFS` file system-specific options are only evaluated for the *special* `fcdisk` entry.

  File system-specific options can be used to have the `devFS` file system assign disks that appear on one port to another port. This option should only be used for connecting a `FC600E` disk cabinet.

  The following options are available:

  `remount`
  : Can be used to notify a mounted file system of another allocation table.

sport*XX*=sport*YY*
: Specifies the allocation table. All disks connected to the port(s) to the right of the equals sign, should be addressed under the name of the port to the left of the equals sign.

WARNING

`mount` does not check when using an allocation table whether the specified assignment is physically feasible.

The data may be corrupted if an allocation table is used without installing the `SImpd` MultiPath package at the same time.

NOTES

If the directory on which the file system is to be mounted is a symbolic reference, the file system is mounted on the directory to which the symbolic reference refers, rather than on top of the symbolic link itself.

EXAMPLES

`FC600E` disk cabinets are generally connected to a host via two ports. The disks in a disk cabinet should be accessed from applications via two different port-compliant device nodes. If one of the ports is not available, the disk can no longer be accessed via this port name. With an allocation table, the disk can be accessed on both ports with the same name.

It is assumed that a `FC600E` disk cabinet is connected to `sport.06` and `sport.07`. Partition "s0" of a disk with the FC loop id 5 can be accessed under the names `sdisk.06.05s0` and `sdisk.07.05s0`. If port `sport.06` fails, the disk can no longer be accessed under `sdisk.06.05s0` but still under `sdisk.07.05s0`.

The `devFS` file system allows `sdisk.07.05s0` to be accessed under the name `sdisk.06.05s0` on the basis of an allocation table. The `devFS` file system must be informed of the required allocation with

```
mount –F devfs –o sport.06=sport.07
```

All disks accessible as `sdisk.07.` are then only accessible under `sdisk.06.`

If a further `FC600E` disk cabinet is connected on the system to `sport.14` and `sport.15`, the allocation table can be extended. The `mount` call must then look as follows:

```
mount –F devfs –o sport.06=sport.07,sport.14=sport.15
```

A complete `mount` call may look as follows:

```
mount –F devfs –o sport.06=sport.07,sport.14=sport.15 /fcdisk /dev/fcdisk
```

The complete name of the device node for accessing partition "s0" of a disk with the loop id 5 would be `/dev/fcdisk/sdisk.06.05s0` or `/dev/fcdisk/sdisk.14.05s0`.

The device nodes `/dev/fcdisk/sdisk.07.05s0` or `/dev/fcdisk/sdisk.14.05s0` no longer exist!

Attention

A `mount` call

```
mount –F devfs –o sport.06=sport.07=sport.14=sport.15
```

gives rise to an allocation table, where the ports `sport.07`, `sport.14` and `sport.15` are mapped to `sport.06`.

Names of devices that can be changed via an allocation table appear a number of times under *mount_point*.

If `sdisk.06.05s0` can be accessed both via `sport.06` and `sport.07`, the name `sdisk.06.05s0` appears twice under `/dev/fcdisk/`.

FILES

`/etc/mnttab`
: list of file systems currently mounted

`/etc/vfstab`
: table of automatically mounted resources

SEE ALSO

mount(1M), mkdir(2), mount(2), open(2), umount(2), mnttab(4), vfstab(4), devfs(7), vdisklite(7), autoconf(8).

System Administrator's Reference Manual

mount(1M-ufs)

NAME

`mount(ufs)` - mount `ufs` file systems

SYNOPSIS

`mount` [`-F ufs`] [*generic_options*] [`-o` *specific_options*] {*special* | *mount_point*}

`mount` [`-F ufs`] [*generic_options*] [`-o` *specific_options*] *special mount_point*

DESCRIPTION

`mount` attaches a `ufs` file system to the file system hierarchy at the pathname location *mount_point*, which must already exist. If *mount_point* has any contents prior to the `mount` operation, these remain hidden until the file system is once again unmounted.

OPTIONS

The `-F ufs` option specifies the `ufs` FSType.

The *generic_options* are options supported by the generic `mount` command.

The specific options are:

`-o` Specifies `ufs` file system specific options. If invalid options are specified, a warning message is printed and the invalid options are ignored. The following options are available:

`f` Fakes an `/etc/mnttab` entry, but does not actually mount any file systems. Parameters are not verified.

`n` Mounts the file system without making an entry in `/etc/mnttab`.

`rw|ro`
Read/write or read-only. Default is `rw`.

`nosuid`
By default the file system is mounted with setuid execution allowed. Specifying `nosuid` overrides the default and causes the file system to be mounted with setuid execution disallowed.

`remount`
Used in conjunction with `rw`. A file system mounted read-only can be *remounted* read-write. Fails if the file system is not currently mounted or if the file system is mounted `rw`.

`force`
Used in conjunction with `ro` (or `-r`). A *dirty* file system mounted read-only with this specific option. This is a very dangerous operation and should be avoided by using `fsck` to verify that the file system is consistent. However, it is very useful if used carefully.

NOTES

If the directory on which a file system is to be mounted is a symbolic link, the file system is mounted on the directory to which the symbolic link refers, rather than on top of the symbolic link itself.

LOCALE

The `LC_MESSAGES` environment variable governs the language in which message texts are displayed.

If `LC_MESSAGES` is undefined or is defined as the null string, it defaults to the value of `LANG`. If `LANG` is likewise undefined or null, the system acts as if it were not internationalized.

If any of the locale variables has an invalid value, the system acts as if none of the variables was set.

The `LC_ALL` environment variable governs the entire locale. `LC_ALL` takes precedence over all the other environment variables that affect internationalization.

FILES

`/etc/mnttab`
table of mounted file systems

SEE ALSO

`fsck`(1M), generic `mount`(1M), `mkdir`(2), `mount`(2), `open`(2), `umount`(2), `mnttab`(4).

System Administrator's Reference Manual

mount(1M-vxfs)
VxFS

NAME

`mount(vxfs)` - mount a Veritas File System (VxFS)

SYNOPSIS

`mount` [`-F vxfs`] [*generic_options*] [`-o` *specific_options*] {*special* | *mount_point*}

`mount` [`-F vxfs`] [*generic_options*] [`-o` *specific_options*] *special mount_point*

DESCRIPTION

This manual page is an addendum to the `mount(1M)` manual page. It describes the Veritas File System (VxFS) specific features of the `mount` command.

The `mount` command notifies the system that *special*, a Veritas block special device, is available to users from the *mount_point* which must exist before `mount` is called; it becomes the name of the root of the newly mounted *special*.

OPTIONS

`-F vxfs`
: Specifies a Veritas file system type (*FSType*).

generic_options
: Options supported by the generic `mount` command.

`-o` Specifies the Veritas file system specific options in a comma-separated list. The available options are:

`rw|ro`
: Read/write or read-only. The default is `rw`.

`suid|nosuid`
: Set-user-id is honored or ignored on execution. The default is `suid`.

`log|nolog`
: Enables or disables activity logging. File system integrity across system failure requires that activity logging be enabled. The `nolog` option is used for memory resident or temporary file systems. The default is `log`.

`blkclear`
: Ensures that all data extents are cleared before being allocated to a file (requires synchronous zeroing of certain newly allocated extents).

`remount`
: Used in conjunction with `rw`. A file system mounted read-only can be `remount`ed read-write. `remount` fails if the file system is not currently mounted or if the file system is mounted `rw`.

`snapof=`*snapof_special*
: Mounts the file system as a snapshot of *snapof_special*, where *snapof_special* is the block special file of a mounted Veritas file system.

`snapsize=`*size*
: *size* is the size in sectors of the snapshot file system being mounted. □
Necessary in conjunction with `snapof`.

`mincache=direct|dsync|closesync`
: This option is used to alter the caching behavior of the file system. The `direct` value will cause any writes without the `O_SYNC` flag and all reads to be handled as if the `VX_DIRECT` caching advisory had been set. The `dsync` value will cause any writes without either the `O_SYNC` flag or the `VX_DIRECT` caching advisory to be handled as if the `VX_DSYNC` caching advisory had been set. The `closesync`, `dsync`, and `direct` values all cause the equivalent of an `fsync(2)` to

be run when the file is closed.

`convosync=direct|dsync|closesync`
: This option is used to alter the caching behavior of the file system for `O_SYNC` I/O operations. The `direct` value will cause any reads or writes with the `O_SYNC` flag to be handled as if the `VX_DIRECT` caching advisory had been set instead. The `dsync` value will cause any writes with the `O_SYNC` flag to be handled as if the `VX_DSYNC` caching advisory had been set instead. The `closesync` value delays `O_SYNC` writes so that they do not take effect immediately. The `closesync`, `dsync`, and `direct` values all cause the equivalent of an `fsync(2)` to be run when any file accessed with the `O_SYNC` flag is closed.

`datainlog|nodatainlog`
: Normally, the Veritas file system will do `O_SYNC` writes by logging the date and the time change to the inode (`datainlog`). If the `nodatainlog` option is used, the logging of synchronous writes is disabled. Then, the `O_SYNC` writes will write the data into the file and update the inode synchronously before returning to the user.

`badinode=delete|fsdisable|panic`
: The system administrator uses these options to determine the behavior of the system when it is mounted where an inode is to be marked as "bad".

  `badinode=delete` marks the inode as "bad" (`VX_AF_IBAD`), which means that it will be removed when the next check (`fsck`) is performed (default).

  `badinode=fsdisable` deactivates the file system but does not mark the inode as "bad". A complete file system check is required in this case to resolve the problem.

  `badinode=panic` initiates a system panic; the inode is not marked as "bad".

Only a privileged user can mount file systems.

NOTES

Unlike some file system commands, multiple `–o` options to mount do not accumulate; instead the last one in the command line takes precedence.

FILES

`/etc/mnttab`

SEE ALSO

`mount(1M)`, `setmnt(1M)`, `mount(2)`, `setuid(2)`, VxFS-specific `fs`(4), `mnttab(4)`.

System Administrator's Reference Manual

mountall(1M)

NAME

`mountall, umountall -` mount, unmount multiple file systems

SYNOPSIS

`mountall` [`-F` *FSType*] [`-L` *level*] [`-l` | `-r`] *file_system_table* | `-`

`umountall` [`-F` *FSType*] [`-k`] [`-l` | `-r`]

DESCRIPTION

These commands may only be executed by a privileged user.

`mountall` is used to mount file systems according to a *file_system_table*. (`/etc/vfstab` is the default file system table.) The special file name "-" reads from standard input. If the dash is specified, then the standard input must be in the same format as `/etc/vfstab`. With no arguments `mountall` restricts the mount to all systems with `automnt` field set to `yes` in the *file_system_table*.

Before each file system is mounted, a sanity check is performed using `fsck`(1M) to see if it appears mountable. If the file system does not appear mountable, it is fixed, using `fsck`, before the mount is attempted.

`umountall` causes all mounted file systems except `root`, `/proc`, `/var`, `/dev/fd` and `/usr` to be unmounted. If the *FSType* is specified `mountall` and `umountall` limit their actions to the *FSType* specified.

OPTIONS

`-F` Specifies the file system type to be mounted or unmounted. If *FSType* is specified the action is limited to file systems of this *FSType*.

`-l` Limits the action to local file systems.

`-L` Limits the action to file systems at a specified level. Legal levels are `0` through `9` and `u`:

Level `0`:
: Essential file systems for boot process (such as `/usr`, `/var`)

Level `1-9`:
: Other file systems.

Level `u`:
: `mount` and `fsck` all levels except level 0.

`-r` Limits the action to remote file system types.

`-k` Sends a `SIGKILL` signal to processes that have files opened.

DIAGNOSTICS

No messages are printed if the file systems are mountable and clean.

Error and warning messages come from `fsck`(1M) and `mount`(1M).

SEE ALSO

`fsck`(1M), `fuser`(1M), `mount`(1M), `signal`(2), `vfstab`(4), `mnttab`(4).

System Administrator's Reference Manual

mpcntl(1M)

NAME

`mpcntl` - standard interface for multiprocessing-specific services

SYNOPSIS

`mpcntl` `-b` `-c` *cpu* [`-c`*cpu* ...] {`-p`*pid* [`-p`*pid* ...] | *filename* [*options*]}

`mpcntl` `-x` `-c` *cpu* [`-c`*cpu* ...] {`-p`*pid* [`-p`*pid* ...] | *filename* [*options*]}

`mpcntl` `-u` {`-p`*pid* [`-p`*pid* ...] | *filename* [*options*]}

`mpcntl` `-a` {`-p`*pid* [`-p`*pid* ...] | *filename* [*options*]}

`mpcntl` `-n` {`-p`*pid* [`-p`*pid* ...] | *filename* [*options*]}

`mpcntl` `-f` `-p`*pid* [`-p`*pid* ...]

`mpcntl` `-N`

`mpcntl` `-I` [`-c`*cpu* ...]

`mpcntl` `-B` {e|d} `-c`*cpu* [`-c`*cpu* ...]

`mpcntl` `-X` {e|d} `-c`*cpu* [`-c`*cpu* ...]

`mpcntl` `-O` `-c` *cpu* [`-c`*cpu* ...]

`mpcntl` `-R` `-c` *cpu* [`-c`*cpu* ...]

`mpcntl` `-D` {e|d} `-c`*cpu* [`-c`*cpu* ...]

`mpcntl` `-H` `-c` *cpu* [`-c`*cpu* ...]

`mpcntl` `-A` `-c` *cpu* [`-c`*cpu* ...]

DESCRIPTION

`mpcntl` provides a number of multiprocessor (MP) specific services to application programmers wishing to exploit this environment.

OPTIONS

`-b` Binds a process to the CPU(s) specified by the `-c` option list. The target process(es) is/are specified by the `-p` option in the form of a *pid* list or the target file, *filename*, is `exec`'d with any additional options. The user must either have superuser privileges or have the same effective user ID as the target process and the user must have the special privilege granted [see `pfenable(1M)`].

`-x` Exclusively binds a process to the CPU(s) specified by the `-c` option list. The target process(es) is/are specified by the `-p` option in the form of a *pid* list or the target file, *filename*, is `exec`'d with any additional options. The user must either have superuser privileges or have the same effective user ID as the target process and the user must have the special privilege granted [see `pfenable(1M)`].

`-u` Unbinds (exclusive or regular bindings) a process. The target process(es) is/are specified by the `-p` option in the form of a *pid* list or the target file, *filename*, is `exec`'d with any additional options. The user must either have superuser privileges or have the same effective user ID as the target process.

`-a` Allows a process to gain affinity on CPUs (default condition). See `mpcntl(3X)` for a definition of affinity. The target process(es) is/are specified by the `-p` option in the form of a *pid* list or the target file, *filename*, is `exec`'d with any additional options. The user must either have superuser privileges or have the same effective user ID as the target process.

`-n` Disallows a process from gaining affinity on CPUs. See `mpcntl(3X)` for a definition of affinity. The target process(es) is/are specified by the `-p` option in the form of a *pid* list or the target file, *filename*, is `exec`'d with any additional options. The user must either have superuser privileges or have the same effective user ID as the target process and the user must have the special privilege granted [see `pfenable(1M)`].

`-f` Displays the `mpcntl` flag and binding attributes for the process(es) specified by the `-p` *pid* list. The flag attributes will be: (1) "bind" or "xbnd" or "nbnd" if the process has a normal binding, an exclusive binding or no binding, respectively (2) "naf" or "aff" if the process is not allowed to gain affinity or is allowed to gain affinity, respectively (3) "npr" or "prm" if the process is in non-preemptive mode or preemptive mode, respectively. For "npr" the modifier "[sig]" means that the process is not immune to signals or "[nsig]" means that the process is immune from receiving "caught signals". See `mpcntl(3X)` for a detailed specification of signal behavior with regards to non-preemptive mode.

`-N` Returns the number of CPUs configured in the system. This number does not imply that all CPUs are active at the present time.

`-I` Returns CPU information for the CPUs identified in the `-c` option list. If no target CPUs are specified, then all information for all active CPUs is returned. The following information is provided:

- The CPU physical ID.
- The CPU status flags.
- The CPU attributes.
- The CPU type.
- The FPU type(s).
- The secondary cache size.
- The number of processes bound to the CPU.
- The number of processes exclusively bound to the CPU.
- The number of processes with driver bindings to the CPU.
- The number of processes with interrupt bindings to the CPU.
- The CPU clock rate.

The status flags are interpreted as follows:

`i/I` The CPU is (is not) in service.

`a/A` The CPU is available (is not available) for normal process execution.

`b/B` The CPU is available (is not available) for normal process bindings.

`x/X` The CPU is available (is not available) for exclusive process bindings.

`c/C` The CPU cache is (is not) enabled.

`o/O` The CPU is (is not) accepting I/O interrupts.

The attribute flags are interpreted as follows:

`i/I` The CPU is (is not) capable of taking I/O interrupts.

`a/A` The CPU has (does not have) uniform access to I/O addresses.

`u/U` The CPU has (does not have) uniform access to memory.

`m/M` The CPU is not (is) the master/boot CPU.

`d/D` The CPU has no (has) drivers bound to it.

`-B` Enables/Disables the ability of CPU(s) identified in the `-c` option list to accept normal processes bindings. The "e" modifier implies an enabling of the CPU's ability to accept bindings and the "d" modifier implies a disabling of the CPU's ability to accept bindings. The user must have superuser privileges.

`-X` Enables/Disables the ability of CPU(s) identified in the `-c` option to accept exclusive processes boundings. The "e" modifier implies an enabling of the CPU's ability to accept bindings and the "d" modifier implies a disabling of the CPU's ability to accept bindings. The user must have superuser privileges.

`-O` Removes the specified CPU(s) from service. This has the effect of canceling bindings to the CPU(s) and may have the side effect of making drivers that are bound to the CPU(s) inaccessible. By default,

all configured CPUs are started when the system is booted. *This option may not be supported by all implementations*.

`-D` Enables/Disables the ability of CPU(s) identified in the `-c` option to receive distributed I/O interrupts. The "e" modifier implies an enabling of the CPU's ability to accept interrupts and the "d" modifier implies a disabling of the CPU's ability to accept bindings. The must have superuser privileges and *the option may not be supported by all implementations*.

`-R` Restores the specified CPU(s) to service. CPU(s) that have been removed from service via the `-O` option can be restored to service with this option. Independent of support for the `-O` option, the `-R` option *may not be supported by all implementations*.

`-H` Marks the specified CPU(s) as not available. This implies that the CPU(s) will not attempt to execute processes other than those with kernel bindings for it (e.g. for purposes of supporting a bound driver). The CPU may still vie for interrupts and other global system processing functions (e.g. callout processing). The user must have superuser privileges.

`-A` Restores the availability of the specified CPU(s). This allows the CPU(s) to execute processes, within the constraints of binding, as well as service kernel bound services. Effectively, this option is the inverse of the `-H` option.

SEE ALSO

`launchit`(1M), `pfenable`(1M), `mpcntl`(3X).

System Administrator's Reference Manual

mpdcontrol(1M)

NAME

`mpdcontrol` - control program for the `MultiPath` driver `mpd`

SYNOPSIS

`mpdcontrol` `–o` *action* [`unit` # [`device` #]]

DESCRIPTION

The behavior of the `MultiPath` `mpd` driver can be influenced or settings can be queried with `mpdcontrol`.

OPTIONS

The "#" character below represents a decimal number, which indicates the number of a particular `unit` or a specific `device` in a unit.

The *action* argument need not be filled out fully, it will suffice if the argument can be identified uniquely.

`–o list` [`unit` #]
: Information is displayed about a particular unit or about all units of the `MultiPath` driver.

`–o xlist` [`unit` #]
: Extended information is displayed about a particular unit or about all units of the `MultiPath` driver.

`–o disable unit` # `device` #
: The device # is set to unusable on unit #.

`–o enable unit` # `device` #
: The device # is set to usable on unit #.

`–o rdonly unit` # `device` #
: The device # is set to read only on unit #.

`–o wronly unit` # `device` #
: The device # is set to write only on unit #.

`–o rdwr unit` # `device` #
: The device # is opened for reading and writing on unit #.

`–o backup unit` # `device` #
: The device # is set as a backup on unit #; this means that this device will only be used if no other device is now available on the unit.

`–o policy` [*policy* [`unit` #] [`device` #]]
: Sets a particular access policy for all `MultiPath` units or only for the unit #. If no policy is defined, the possible policies are displayed.

`–o init`
: Initializes the driver. The `MultiPath` driver is only available if this command has been issued.

`–o reinit`
: Reinitializes the driver. The `MultiPath` driver cannot automatically recognize when new devices have been added. New units or new devices for an existing unit can be added with `reinit`. The numbers of the existing units do not change here.

`–o remove`
: Uninstalls the driver. The `MultiPath` devices are then no longer controlled by the `MultiPath` driver.

`–o statis` {`on` | `off`}
: Switches driver statistics on or off, see also `sar(1)` .

`–o hello` [`unit` #]
: The devices in the unit # or all units are addressed temporarily. This option can be used to recognize errors in devices on which no I/O operations are running.

`-o watch`
: If `mpdcontrol` is called with `watch`, `mpdcontrol` creates a new process which is then always informed by the `MultiPath` driver as soon as the driver has established an error on a device in a unit. This process automatically executes the action `hello` on all units.

POLICIES

The following policies are supported:

`pingpong`
: In the case of `pingpong` policy, the jobs are distributed equally to all devices in a unit (round robin).

`latency`
: In the case of `latency` policy, a job is always sent to the device with the lowest latency time.

`toggle`
: In the case of `toggle` policy, a job is sent to the device with the lowest latency time. If several devices have the same latency time, the job is sent to the device that did not receive the last job (latency time + round robin).

`simple`
: In the case of `simple` policy, a job is sent to the first available device. No load distribution takes place!

`prefer`
: In the case of `prefer` policy, a job is sent to the preferred device. No load distribution takes place!

  If no preferred device is specified, `mpdcontrol` calculates the preferred device using the formula (number of unit) modulo (number of devices in unit).

If devices have their own large cache, the `toggle` policy should be selected.

If devices have no cache or just a small cache, the `prefer` policy should be selected.

EXAMPLES

The `mpdcontrol -o list` call shows the `MultiPath` units and the associated devices:

Sample output:

```
Unit Dev# MPD_ID          DeviceName           State
=======================================================
   0    0 LK083760        iof0/sdisk.15.babba  enb prf
        1                 iof0/sdisk.14.babba  enb
-------------------------------------------------------
   1    0 LK083698        iof0/sdisk.15.a16bb  enb
        1                 iof0/sdisk.14.a16bb  enb prf
-------------------------------------------------------
```

More precise information on `MultiPath` unit "0" can be displayed with the call `mpdcontrol -o xlist unit 0`.

Sample output:

```
Unit Dev# MPD_ID/Policy   DeviceName           FC_AL DevMajMin IOcnt State
=============================================================================
   0    0 LK083760        iof0/sdisk.15.babba      4 416,0       93 enb prf
        1 prefer          iof0/sdisk.14.babba      4 416,16       0 enb
-----------------------------------------------------------------------------
```

Device "1" in `MultiPath` unit "0" can be set to `rdonly` with the call `mpdcontrol -o rdonly unit 0 device 1`.

If actions only require one unit as an additional argument, it is also possible to enter the device ID instead of the "`unit` #". This means that the call `mpdcontrol -o xlist LK083760` has the same effect as the call `mpdcontrol -o xlist unit 0`.

In the case of actions that require "`device` #" as an argument in addition to "`unit` #", the `autoconf` name of

the device can be entered instead of "`unit` # `device` #". This means that the call `mpdcontrol –o rdonly iof0/sdisk.babba` has the same effect as the call `mpdcontrol –o rdonly unit 0 device 1`.

NOTES

When initialized, the `mpd` driver searches the `autoconf` tree for available `MultiPath` devices. If a device is found, the `MultiPath` driver enters itself in internal system `autoconf` entries for this device. These internal system `autoconf` entries are evaluated by the device file system [cf. `devfs`(7)]. `MultiPath` units are only accessible via the device file system.

If the hardware configuration of a machine changes, the number of a unit (`unit` #) of the `MultiPath` units may change following a reboot.

New `MultiPath` units are added to the rear while the `MultiPath` driver is active.

SEE ALSO

`devfs`(7) , `autoconf`(8) .

System Administrator's Reference Manual

mpstat(1M)

NAME

`mpstat -` report multi-processor statistics

SYNOPSIS

`/usr/ucb/mpstat` [*interval*] [*count*]

DESCRIPTION

`mpstat` reports certain statistics kept about process, virtual memory, exception, and CPU activity. It provides a subset of the virtual memory information provided by `vmstat`, but expands the CPU utilization information to show the activity on each of the processors on the system. `vmstat` gives the composite CPU statistics for all processors on the system. `mpstat` reports in the first line a summary of the system activity since the system has been booted.

If *interval* is specified, then successive lines are reported over the last *interval* seconds. For example, `mpstat 5` will print what the system is doing every five seconds. This is a good choice for printing interval because some statistics are sampled every five seconds in the system. Watch the output for a while will to identify what statistics are sampled in what interval.

If a *count* is given, the statistics are repeated *count* times.

These format fields are available:

`procs`
Information about numbers of processes in various states.

- `r` In run queue.
- `b` Blocked for resources (I/O or paging).
- `w` Operable process or short sleeper (< 20 secs) but swapped.

`memory`
Information about the usage of virtual and real memory. Virtual pages are considered active if they belong to processes which are running or have run in the last 20 seconds. A page here is 1024 bytes.

- `avm` Active virtual pages.
- `fre` Size of the free list.

`page` Information about page faults and paging activity. These are averaged each five seconds, and given in units per second.

- `pi` Pages paged in.
- `po` Pages paged out.
- `si` Pages swapped in.
- `so` Pages swapped out.

`Cpu` Information about exceptions and interrupts. These are given in units per second. Breakdown of CPU activity (one set per CPU).

- `int` Device interrupts fielded by this CPU.
- `cint` Inter-CPU interrupts fielded by this CPU.
- `fint` Floating point exceptions fielded by this CPU.
- `syc` System calls per second.
- `cs` Cpu context switch rate (switches/sec).
- `id` Percentage of CPU time spent in idle or I/O wait mode.
- `us` Percentage of CPU time spent in user mode.
- `sy` Percentage of CPU time spent in system mode (other than idle).

`io` Percentage of CPU time spent in I/O wait mode.

SEE ALSO

`vmstat`(1M).

System Administrator's Reference Manual

mux(1M)

NAME

`mux` - control connection to a cluster terminal

SYNOPSIS

`/sbin/mux` *real_terminal pseudo_terminal* [*options*]

`/sbin/mux` *real_terminal getty_type*

DESCRIPTION

The `mux` command handles the protocol used to communicate with a cluster terminal which is connected to a Reliant UNIX host. It allows the multiplexing of a terminal and a second device on the same line.

There are two distinct modes of operation. The first is the preferred method as it gives greater control to the operation of `mux`. In the first mode, the `mux` process has at least two parameters. The first is the line to which the cluster terminal is connected, whereas the second is the name of the pseudo device through which the user process is to be connected. Any additional options specified, control the various parameters used by `mux` (see the **OPTIONS** section for more details). With this mode, `mux` is usually started at boot time from SAC (Service Access Controller).

The second mode of operation is only available if the `mux` process has been compiled to support it. If not, then an error message is generated and `mux` be terminated.

In this mode, the first parameter is the name of the line to which the cluster terminal is connected, whereas the second parameter is a single letter defining one of a set of options. These options define the operation of the `mux` process. In this mode, `mux` also starts a `getty` process on the pseudo terminal that is created.

In this mode, `mux` would normally be started from `/etc/init`. This mode is for backwards compatibility to an older version of `mux`.

The second parameter can have one of the following values. If the second parameter does not come from this list, the *other* parameters are used, and the second parameter is passed directly to the `getty` when it is started.

Parameter	Speed	Protocol	Window_size	Nchans	getty_parm
A	38400	No	2	1	N
E	38400	Yes	2	2	N
C	38400	No	2	1	N
D	38400	Yes	2	2	N
a	1200	No	2	1	h1200
b	1200	Yes	2	2	h1200
c	2400	No	2	1	h2400
d	2400	Yes	2	2	h2400
e	9600	No	2	1	h
g	9600	Yes	2	2	h
z	3840	No	8	1	S

	0				
Z	38400	Yes	8	2	S
other	38400	Yes	8	2	*other*

No other options are available in this mode. If the number of channels is greater than one, `mux` opens `/etc/cluster` and looks for an entry for the given real device. If one is found, the second channel is configured using the information provided in `/etc/cluster`; otherwise, no second channel is configured.

OPTIONS

In the first mode of operation, the following options can be specified:

`-C` *device*
: This informs `mux` that the second device is a chip card terminal which is to have the name given by the *device* parameter. A parameter must be supplied.

`-D` *device*
: This informs `mux` that the second device is a *printer* which is to have the name given by the *device* parameter. A parameter must be supplied.

`-B` *device*
: This informs `mux` that the second device is another terminal which is to have the name given by the *device* parameter. A parameter must be supplied.

`-S` *device*
: This informs `mux` that the second device is a special device with unknown characteristics. The name of the device is given by the *device* parameter. A parameter must be supplied.

`-G` *speed*
: This option sets the speed of the real terminal line to be speed. Values currently supported are:

  `1200`, `2400`, `9600`, `38400`

`-L`
: This option forces the `mux` process to use large window sizes and a large buffer space for increased performance. This option does not work on some cluster terminals.

`-N`
: By default, the `mux` process uses a multiplexing protocol to communicate with the cluster terminal. This flag forces this protocol off so that the terminal works in basic mode. In this mode, no additional devices may be connected.

  The specification of a second device forces the use of a multiplexing protocol.

`-T` *params*
: This option is used for controlling the optional trace package. A special version of the `mux` process, which includes a tracing facility, must be available for this option to take effect. It is for the use of `mux` maintainers only.

FILES

`/etc/cluster`

mvdir(1M)

NAME

`mvdir` - move a directory

SYNOPSIS

`/usr/sbin/mvdir` *dirname name*

DESCRIPTION

`mvdir` moves directories within a file system. *dirname* must be a directory. If *name* does not exist, it will be created as a directory. If *name* does exist, and is a directory, *dirname* will be created as *name/dirname*. *dirname* and *name* may not be on the same path; that is, one may not be subordinate to the other. For example:

`mvdir x/y x/z`

is legal, but

`mvdir x/y x/y/z`

is not legal.

LOCALE

The `LC_MESSAGES` environment variable governs the language in which message texts are displayed.

If `LC_MESSAGES` is undefined or is defined as the null string, it defaults to the value of `LANG`. If `LANG` is likewise undefined or null, the system acts as if it were not internationalized.

If any of the locale variables has an invalid value, the system acts as if none of the variables was set.

The `LC_ALL` environment variable governs the entire locale. `LC_ALL` takes precedence over all the other environment variables that affect internationalization.

NOTE

Only the superuser can use `mvdir`.

SEE ALSO

`mkdir(1)`, `mv(1)`.

ncheck(1M) - osxfsrestore(1M)

ncheck(1M)

NAME

`ncheck(generic)` - generate a list of path names versus i-numbers

SYNOPSIS

`ncheck` [`–F` *FSType*] [`–Vas`] [`–i`*i-numbers*] [`–o` *specific_options*] [*special ...*]

DESCRIPTION

`ncheck` with no options generates a pathname versus i-number list of all files on *special*. If *special* is not specified on the command line, the list is generated for all *specials* in `/etc/vfstab` for entries which have a numeric *fsckpass*. *special* is a block special device or a raw device on which the file system exists.

There are two types of option: *generic_options* and *specific_options*. *generic_options* indicate options that are supported not only by the generic but also by the *FSType*-specific module of the command (if that command exists). *specific_options* indicate suboptions that are supported only by the *FSType*-specific module of the command. They are specified in a comma-separated list and/or as keyword-attribute pairs for interpretation by

the *FSType*-specific module of the command. These suboptions are specified via the –o option.

The *generic_options* are:

–F Specifies the *FSType* on which to operate. The *FSType* should either be specified here or be determinable from /etc/vfstab by finding an entry in the table that has a numeric *fsckpass* field and a matching *special* if specified.

–V Echoes the complete command line, but does not execute the command. The command line is generated by using the options and arguments provided by the user and adding to them information derived from /etc/vfstab. This option should be used to verify and validate the command line. When used with the *FSType* option, it passes the –V to the *FSType*-specific module for interpretation.

–i *i-number*
Limits the report to those files whose i-numbers follow. The *i-number*s must be separated by commas without spaces.

–a Allows printing of the names "." and "..", which are ordinarily suppressed.

–s Limits the report to special files and files with set-user-ID mode. This option may be used to detect violations of security policy.

–o Used to specify *FSType* specific options if any.

NOTE

This command may not be supported for all *FSTypes*.

FILES

/etc/vfstab
list of default parameters for each file system

SEE ALSO

ff(1M), vfstab(4).
Manual pages for the *FSType*-specific modules of ncheck.

System Administrator's Reference Manual

ncheck(1M-ufs)

NAME

`ncheck(ufs)` - generate pathnames versus i-numbers for ufsfile systems

SYNOPSIS

`ncheck [ –F ufs ] [` *generic_options* `] [ –o m ] [` *special ...* `]`

DESCRIPTION

`ncheck` generates a pathname versus i-number list of all files on the specified *special* device(s) which should represent a `ufs` file system. Names of directory files are followed by "`/.`".

OPTIONS

`–F ufs`
: Specifies the `ufs`-FSType.

generic_options
: Options supported by the generic `ncheck` command.

`–o`
: Specifies `ufs` file system specific options. The available option is:

  `m` Prints mode information.

DIAGNOSTICS

If the file system structure is not consistent, "`??`" denotes the parent of a parentless file and a pathname beginning with "`. . .`" denotes a loop.

SEE ALSO

generic `ncheck(1M)` .

System Administrator's Reference Manual

ncheck(1M-vxfs)
VxFS

NAME

`ncheck(vxfs)` - generate path names from inode numbers fora Veritas File System (VxFS)

SYNOPSIS

`ncheck` [`–F vxfs`] [*generic_options*] [`–o m,b=`*blk#*] *special* ...

DESCRIPTION

This manual page is an addendum to the `ncheck(1M)` manual page. It describes the Veritas File System (VxFS) specific features of the `ncheck` command.

`ncheck` generates a path name versus inode number list of files for the Veritas file system. Names of directory files are followed by "`.`".

OPTIONS

`–F vxfs`
: Specifies the Veritas file system type (*FSType*).

generic_options
: Options supported by the generic `ncheck` command.

`–o`
: Specifies Veritas file system specific options. The available options are:

`m`
: Prints mode information.

`b=`*blk#*
: Prints path name containing block number *blk#*.

DIAGNOSTICS

When the file system structure is improper, "`???`" denotes the "parent" of a parentless file and a path name beginning with "`...`" denotes a loop. A path name beginning with "`***`" denotes a directory entry whose nominal "`..`" entry is not in accord with the directory in which it was found.

SEE ALSO

`sort(1)`, `fsck(1M)`, `ncheck(1M)`, `fs(4-vxfs)`.

System Administrator's Reference Manual

ndx(1M)

NAME

ndx - create a subject-page index for a document

SYNOPSIS

ndx *subjfile* "*formatter command line*"

DESCRIPTION

ndx, given a list of subjects (*subjfile*), searches a specified document and writes a subject-page index to the standard output.

subjfile is the list of subjects to be included in the index. Each subject must begin on a new line and have the following format:

word1 [*word2* ...] [, *wordk* ...]

For example,

printed circuit boards
arrays
arrays, dynamic storage
Smith, W. P.
printed circuit boards, channel-oriented multi-layer
Aranoff
University of Illinois
PL/1

The subject must start in column 1.

The syntax for the formatter command line is

formatter [*option(s)*] *file*(*s*)

It is the command that will be used to create the final form of the document. The following are examples of valid formatter command lines:

```
mm  −Tlp  file(s)
nroff  −mm  −Tlp  −rW60  file(s)
troff  −rB2  −Taps  −r01.5i  file(s)
```

For more information on the formatter command line, see mm(1M), mmt(1M), nroff(1M) and troff(1M).

The document must include formatting commands for mm, nroff, or troff. The formatter command line tells ndx whether troff, nroff, mm or mmt would be used to produce the final version of the document. troff or mmt specifies troff as the formatting program; nroff or mm specifies nroff as the formatting program. The *option*(*s*) are those that would be given to the troff, nroff, mm or mmt command in printing the final form of the document, and are necessary to determine the correct page numbers for subjects as they are located in the document.

ndx does not actually cause the final version of the document to be printed. The author must create the document separately. The indexer, of course, should not be used until the document is complete and no further changes are expected.

EXAMPLES

The command

ndx *subjfile* "nroff −mm −rW70 *file*(*s*)" >*indexfile*

would produce a subject-page index for the document *file*(*s*) and take its subjects from the list, *subjfile*. The

page numbers would correspond to the document produced by

`nroff –mm –rW70` *file*(*s*)

The command

`ndx` *subjfile* `"mm –rW60 –rN2 –rO0 ch1 ch2 ch3" >`*indexfile*

would produce a subject-page index for the documents ch1, ch2, and ch3. The page numbers would correspond to the documents produced by

`mm –rW60 –rN2 –rO0 ch1 ch2 ch3`

The command

`ndx` *subjfile* `"troff –rB2 –rW5i –rO1.5i –mm`*file*(*s*)`" >` *indexfile*

would produce a subject-page index for the document *file*(*s*), and the page numbers would correspond to the document produced by

`troff –rB2 –rW5i –rO1.5i –mm` *file*(*s*)

SEE ALSO

`mm`(1M), `mmt`(1M), `nroff`(1M), `subj`(1M), `troff`(1M).

System Administrator's Reference Manual

neqn(1M)

NAME

neqn - format mathematical text for nroff

SYNOPSIS

neqn [−d*xy*] [−p*n*] [−s*n*] [−f*n*] [−] *file* ...

DESCRIPTION

neqn is a preprocessor for formatting mathematical text with nroff on typewriter-like terminals. Normal usage is:

neqn [*options*] *file* ... | nroff [*options*] | [*printer*]

If you do not specify files (or if you specify − as the last argument), neqn reads from standard input.

Full details of use are given in eqn(1M) and in the **REFERENCE** cited below.

SEE ALSO

eqn(1M) , nroff(1M) , eqnchar(5) .

REFERENCE

B. W. Kernighan, L. L. Cherry, *Typesetting Mathematics — User's Guide* (2nd Ed.)

System Administrator's Reference Manual

newaliases(1M)

NAME

`newaliases -` rebuild the database for the mail aliases file

SYNOPSIS

`newaliases`

DESCRIPTION

`newaliases` rebuilds the random access database for the mail aliases file `/etc/aliases`. It must be run each time this file is changed in order for the change to take effect.

`newaliases` is identical to "`sendmail –bi`".

DIAGNOSTICS

The `newaliases` utility exits `0` on success, and `>0` if an error occurs.

FILES

`/etc/aliases`
: the mail aliases file

SEE ALSO

`sendmail`(1M), `aliases`(4).

System Administrator's Reference Manual

newfs(1M)

NAME

`newfs` - construct a new file system

SYNOPSIS

`/sbin/newfs` [`–N`] [`–v`] [*mkfs_options*] *special*

DESCRIPTION

`newfs` is a "friendly" front-end to the `mkfs(1M)` program. `newfs` will gather information on the type of disk a file system is being created, and calculate the appropriate parameters to use in calling `mkfs`. It then builds the file system by forking `mkfs` and `fsirand`. *special* must be a special character device file (for example, `/dev/ios0/rsdisk000s1`).

OPTIONS

These options override the default actions taken by `newfs`:

`–N` Causes the file system parameters to be printed out without actually creating the file system.

`–v` Causes `newfs` to print out its actions, including the parameters passed to `mkfs`.

The file system specific *mkfs_options* used to override default parameters passed to `mkfs` are:

`–f` *fragsize*
: The fragment size of the file system in bytes.

`–b` *bsize*
: The block size of the file system in bytes.

`–m` *minfree*
: The percentage of space reserved from normal users; the minimum free space threshold. The default value used is 10%.

`–c` *cgsize*
: The number of cylinders per cylinder group in a file system. The default value used is 16.

`–i` *nbpi*
: This specifies the density of inodes in the file system. The default is to try to create an inode for each 2048 bytes of data space. If fewer inodes are desired, a larger number should be used; to create more inodes, a smaller number should be given. See `mkfs(1M)` for further information.

`–o` *opt*
: The optimization for the file system. It can be either `space` or `time`.

The disk drive specific *mkfs_options* used to override default parameters passed to `mkfs` are:

`–s` *nsect*
: The size of the file system in sectors.

`–t` *ntrack*
: The number of a tracks per cylinder on the disk.

`–r` *rpm*
: The speed of the disk in revolutions per minute (normally 3600).

`–d` *gap*
: The rotational delay (or gap) for the file system.

`–a` *apc*
: The number of alternate sectors per cylinder.

`–S` *secsize*
: The sector size derived from the device during initial drive format. The default for this is 512 bytes for SCSI drives.

`-q` This causes the `mkfs -f` option to be set. This causes `mkfs` to skip the 10 second delay.

NOTES

This command may not be supported for all *FSTypes*.

FILES

`/etc/vfstab`
list of default parameters for each file system

SEE ALSO

`fsirand`(1), `vfstab`(4).
Manual pages for the *FSType*-specific modules of `mkfs`.

System Administrator's Reference Manual

nroff(1M)

NAME

`nroff` - text formatting language

SYNOPSIS

`nroff` [*option* ...] *file* ... | *printer*

DESCRIPTION

`nroff` formats text contained in *file*(*s*) (standard input by default) for printing on typewriter-like devices and line printers.

An argument consisting of a minus (–) is taken to be a file name corresponding to the standard input.

OPTIONS

The *option*(*s*), which may appear in any order, but must appear before the *file*(*s*), are:

–o*list*
: Print only pages whose page numbers appear in the *list* of numbers and ranges, separated by commas. A range *N* – *M* means pages *N* through *M*; an initial –*N* means from the beginning to page *N*; and a final *N*– means from *N* to the end. (See **NOTES** below.)

–n*N* Number first generated page *N*.

–s*N* Stop every *N* pages. `nroff` will halt *after* every *N* pages (default *N* = 1) to allow paper loading or changing, and will resume upon receipt of a line-feed or new-line [new-lines do not work in pipelines, e.g. with `mm`(1M)]. This option does not work if the output of `nroff` is piped through `col`(1) . When `nroff` halts between pages, an ASCII BEL is sent to the terminal.

–r*aN* Set register *a* (which must have a one-character name) to *N*.

–i Read standard input after *files* are exhausted.

–q Invoke the simultaneous input-output mode of the `.rd` request.

–Q Messages generated by `.tm` (terminal message) requests will be suppressed.

–z Print only messages generated by `.tm` (terminal message) requests.

–m*name*
: Prepend to the input *file*(*s*) the macro file `/usr/lib/tmac/tmac.`*name*.

–T*tty_type*
: Prepare output for specified terminal. Known *tty_type*(*s*) are:

`2631` Hewlett-Packard 2631 printer in regular mode

`2631-c`
: Hewlett-Packard 2631 printer in compressed mode

`2631-e`
: Hewlett-Packard 2631 printer in expanded mode

`300` DASI-300 printer

`300-12`
: DASI-300 terminal set to 12-pitch (12 characters per inch)

`300s` DASI-300s printer (300S is a synonym)

`300s-12`
: DASI-300s printer set to 12-pitch (12 characters per inch) (300S-12 is a synonym)

`37` TELETYPE Model 37 terminal (default)

`382` DTC-382

`4000a`

Trendata 4000a terminal (4000A is a synonym)

`450` DASI-450 (Diablo Hyterm) printer

`450-12`
DASI-450 terminal set to 12-pitch (12 characters per inch)

`832` Anderson Jacobson 832 terminal

`8510` C.ITOH printer

`lp` generic name for printers that can underline and tab (All text using reverse line feeds, such as those having tables, that is sent to `lp` must be processed with `col`.)

`tn300`
GE Terminet 300 terminal

`X` printers equipped with TX print train

`X97` Xerox 9700 page printer (use with `x9700` filter)

`X97ni`
Xerox 9700 page printer with underscores (no italics) (use with `x9700` filter)

`-e` Produce equally-spaced words in adjusted lines, using the full resolution of the particular terminal.

`-h` Use output tabs during horizontal spacing to speed output and reduce output character count. Tab settings are assumed to be every 8 nominal character widths.

`-u`*n* Set the emboldening factor (number of character overstrikes) for the third font position (bold) to *n*, or to zero if *n* is missing.

NOTES

When `nroff` is used with the `-o`*list* option inside a pipeline [e.g. with one or more of `eqn`(1M) , and `tbl`(1M)], it may cause a harmless "broken pipe" diagnostic if the last page of the document is not specified in *list*.

FILES

`/usr/lib/tmac/tmac.*`
pointers to standard macro files

`/usr/lib/macros/*`
standard macro files

`/usr/lib/nterm/*`
terminal driving tables for `nroff`

`/usr/pub/terminals`
list of supported terminals

SEE ALSO

`col`(1) , `mm`(1M) , `neqn`(1M) , `troff`(1M) , `nterm`(5) .

REFERENCE

J. F. Ossanna, *Nroff/Troff User's Manual*

D. Dougherty, T. O'Reilly, *UNIX TEXT PROCESSING*, Hayden Books, Indianapolis, Indiana 46268 USA, ISBN 0-672-46291-5

System Administrator's Reference Manual

osm(1M)

NAME

`osm, S09osm -` start/stop OSM osmd log daemon

SYNOPSIS

```
/etc/init.d/osm start | stop
/etc/rc2.d/S09osm start | stop
```

DESCRIPTION

The `/etc/rc2.d/S09osm` or `/etc/init.d/osm` script starts the OSM `osmd(1M)` log daemon (OSM – Operating System Messages) when multiuser mode is initiated.

PARAMETERS

`start`
: This parameter starts the `osmd` daemon if it is not already active.

`stop` This parameter stops all `osmd` daemons.

SEE ALSO

`osmd(1M)` .

System Administrator's Reference Manual

osmd(1M)

NAME

`osmd` - log OSM data

SYNOPSIS

`osmd [-d][-I][-T][-O][-h][-o` *name*`][-c` *name*`][-s`*size*`][-m` *max*`][-f` *name*`][-p` *name*`]`

DESCRIPTION

The `osmd` daemon (OSM – Operating System Messages) logs a stream of messages from a device (e.g. `/dev/osm`), file or pipe or from standard input. In the process, it checks the size of the protocol files to prevent the file system containing these files from overflowing.

The `osmd` daemon creates a cycle of three files, by default, under `/etc/.osm`[`.`*MM-DD-hh:mm*], whereby the maximum size of each file is 5 MB. The messages are always written to the file without the suffix. If the maximum size of this file will be exceeded by the next entry, the file is assigned a suffix made up of date and time and a new file is created without the suffix. The messages are written to this new file. If the number of files in the cycle will consequently be exceeded, the oldest files are deleted (generally one file unless the number of files belonging to the cycle has been reduced). The defaults for the `osmd` daemon can be overridden by values from the `/etc/default/osm.conf` configuration file [see `osm.conf(4)`]. The signal 1 (`SIGHUP`) must be sent to the `osmd` daemon to enable it to load the modified content of the `/etc/default/osm.conf` configuration file. The values assigned with the options have the highest priority.

The `osmd` daemon is started with the `/etc/rc2.d/S09osm` script for logging the OSM data when multiuser mode is initiated. The console ring buffer is read from the beginning and further messages are awaited.

The `osmd` daemon takes account of the `/etc/default/osm.filter` filter when reading the messages, provided this filter exists. The filter can contain one regular expression per line against which the messages read are compared. Each regular expression must be enclosed by terminator characters (e.g.: `/^$/` implies a blank line). If "–" is the only character to appear in a line, this means that all messages containing one of the regular expressions are ignored. This is the default . If "\+" is the only character to appear in a line, this means that only lines that contain at least one of the regular expressions from the filter are logged. The default is for no filter to be supplied, i.e. the messages read by `/dev/osm` are not filtered.

OPTIONS

- `-d` Activate test aids.
- `-I` Standard input is read instead of `/dev/osm` (overwrites the `-o` option).
- `-T` Logging (not to standard output) with leading time stamps in the form "*MMM DD hh:mm:ss*" (example: May 22 16:57:51).
- `-O` Additional output to standard output.
- `-h` Output instructions for use.
- `-o` Specify an alternative source for messages to be logged (default: `/dev/osm`).
- `-c` Specify an alternative configuration file (default: `/etc/default/osm.conf`).
- `-s` Specify an alternative maximum size for the protocol files in bytes (range: 10000 bytes – `ulimit` value; default: 5 MB = 5120000 bytes).
- `-m` Specify an alternative number of files that are to form a cycle (range: 2 – 100; default: 3).
- `-f` Specify an alternative filter (default: `/etc/default/osm.filter`).
- `-p` Specify alternative log files (default: `/etc/.osm`[`.`*MM-DD-hh:mm*]).

FILES

`/dev/osm`
: console ring buffer for kernel and driver messages

`/etc/default/osm.conf`
: configuration file for the `osmd` daemon

`/etc/default/osm.filter`
: filter file

`/etc/.osm`[`.`*MM*`-`*DD*`-`*hh*`:`*mm*]
: protocol files

`/var/adm/log/osm`[`.`*MM*`-`*DD*`-`*hh*`:`*mm*]

SEE ALSO

`osm`(1M), `osm.conf`(4), `osm`(7).

System Administrator's Reference Manual

osxfsrestore(1M)

NAME

`osxfsrestore -` incremental file system restore forOSx (TOS) dumptapes

SYNOPSIS

`/sbin/osxfsrestore` *options V* [*filename ...*]

DESCRIPTION

`osxfsrestore` is identical to `ufsrestore` except that it works with dump tapes created by the `dump` command of the OSx Operating System (= TOS on Targon□/35). `osxfsrestore` restores files from these backup tapes thus migrating the file system from OSx to Reliant UNIX.

Since the `ufs` and `osxfs` file systems and the OSx and Reliant UNIX operating systems are different, there may be OSx files that have no meaning on a Reliant UNIX system (for instance, conditional symbolic links have no meaning in Reliant UNIX). These files (and links to these files) will not be extracted. See the diagnostics section below for a list of these file types.

options and any remaining arguments are identical to those of `ufsrestore`. Additionally, if the `V` option is given, then messages describing OSx specific files will be printed along with a note stating that they will not be extracted.

NOTES

All notes applicable to `ufsrestore` apply to `osxfsrestore`, including the following:

`osxfsrestore` will get confused when a restore is tried using a `ufs` dump tape.

DIAGNOSTICS

All diagnostics applicable to `ufsrestore` apply to `osxfsrestore`, including the following:

`Converting from OSx file system format to UFS format.`
: This message will always be printed when running `osxfsrestore`.

.nf

`skipped symbolic link to OSx file` *name* `(ino` *NNN*`) -> symbolic_link_path_name`
`skipped hard link to OSx file` *name* `(ino` *NNN*`) -> path_name_of_link`
: .fi These messages will be printed if there was a link to an OSx specific file that was not extracted.

`skipped OSx dependent file` *name* `(ino` *NNN*`):`
: This message indicates a file that was not extracted because it is OSx specific. It will be followed by one of the following messages which further describes the file.

`OSx conditional symbolic link (->` *symbolic_link_path_name*`).`
`OSx zero length conditional symbolic link.`
`OSx special file (dev = 0x`*NNN*`).`
`OSx auxiliary inodes (security ACLs) removed.`
`OSx bad sector inodes removed.`
: `[NOTE: run format(8) to remap bad disk sectors ].`

FILES

`/dev/rmt8`
: the default tape drive

`/tmp/rstdir*`
: file containing directories on the tape

`/tmp/rstmode*`
: owner, mode, and timestamps for directories

`./restoresymtable`

information passed between incremental restores

SEE ALSO

ufsdump(1M), ufsrestore(1M), format(8).

passmgmt(1M) - pwconv(1M)

passmgmt(1M)

NAME

passmgmt - manage password files

SYNOPSIS

passmgmt –a *options name*

passmgmt –m *options name*

passmgmt –d *name*

DESCRIPTION

The passmgmt command updates information in the password files. This command works with both /etc/passwd and /etc/shadow.

passmgmt –a adds an entry for user *name* to the password files. This command does not create any directory for the new user and the new login remains locked (with the string *LK* in the password field) until the passwd(1) command is executed to set the password.

passmgmt –m modifies the entry for user *name* in the password files. The name field in the /etc/shadow entry and all the fields (except the password field) in the /etc/passwd entry can be modified by this command. Only fields entered on the command line will be modified.

passmgmt –d deletes the entry for user *name* from the password files. It will not remove any files that the user owns on the system; they must be removed manually.

OPTIONS

–c *comment*
: A short description of the login. It is limited to a maximum of 128 characters and defaults to an empty field.

–e *expire*
: The date on which a login can no longer be used; after this date, no user will be able to access this login. (This option is useful for creating temporary logins.) You may type the value of the argument *expire* (which is a date) in various formats (included in /etc/datemsk). For example, you may enter 10/6/90 or October 6, 1990. A value of "" defeats the status of the expired date.

–h *homedir*
: Home directory of *name*. It is limited to a maximum of 256 characters and defaults to /usr/*name*.

–u *uid*
: UID of the *name*. This number must range from 0 to the maximum non-negative value for the system. It defaults to the next available UID greater than 99. Without the –o option, it enforces the uniqueness of a UID.

–o
: This option allows a UID to be non-unique. It is used only with the –u option.

–g *gid*
: GID of the *name*. This number must range from 0 to the maximum non-negative value for the system. The default is 1.

–s *shell*

Login shell for *name*. It should be the full pathname of the program that will be executed when the user logs in. The maximum size of *shell* is 256 characters. The default is for this field to be empty and to be interpreted as `/usr/bin/sh`.

`-l` *new_logname*
: This option changes the *name* to *new_logname*. It is used only with the `-m` option.

`-A` [*operator*][*event*][,...]
: A list of event types or classes separated by a comma. The user's audit event mask is set (`-a` option) or modified (`-m` option). The following operators are available: `-` for deleting the events from the user's audit event mask , `+` for adding the events to the user's audit event mask, `!` for auditing all events with the exception of the listed events. If no operator is specified, the listed events are set in the user's audit event mask. This option is only valid if the (auditing subsystem) is installed.

The total size of each login entry is limited to a maximum of 511 bytes in each of the password files.

DIAGNOSTICS

The `passmgmt` command exits with one of the following values:

`0` = Success.

`1` = Permission denied.

`2` = Invalid command syntax. Usage message of the `passmgmt` command will be displayed.

`3` = Invalid argument provided to option.

`4` = UID in use.

`5` = Inconsistent password files (e.g. *name* is in the `/etc/passwd` file and not in the `/etc/shadow` file, or vice versa).

`6` = Unexpected failure. Password files unchanged.

`7` = Unexpected failure. Password file(s) missing.

`8` = Password file(s) busy. Try again later.

`9` = *name* does not exist (if `-m` or `-d` is specified), already exists (if `-a` is specified), or *new_logname* already exists (if `-m -l` is specified).

LOCALE

The `LC_MESSAGES` environment variable governs the language in which message texts are displayed.

If `LC_MESSAGES` is undefined or is defined as the null string, it defaults to the value of `LANG`. If `LANG` is likewise undefined or null, the system acts as if it were not internationalized.

If any of the locale variables has an invalid value, the system acts as if none of the variables was set.

The `LC_ALL` environment variable governs the entire locale. `LC_ALL` takes precedence over all the other environment variables that affect internationalization.

NOTES

You cannot use a colon or carriage return as part of an argument because it is interpreted as a field separator in the password file.

FILES

`/etc/passwd`

`/etc/shadow`

`/etc/opasswd`

`/etc/oshadow`

SEE ALSO

`passwd(1)`, `useradd(1M)`, `userdel(1M)`, `usermod(1M)`, `default_passwd(4)`, `passwd(4)`, `shadow(4)`.

System Administrator's Reference Manual

pfdisable(1M)

NAME

`pfdisable -` revoke special process flags privileges from users

SYNOPSIS

`/etc/pfdisable –[ ADcCpU ] [` *username ...* `| –E ]`

DESCRIPTION

WARNING:

The use of some of the options offered by this command may severely disrupt the normal functioning of the system.

`pfdisable` revokes the special privileges specified by the option list from *username*, which were granted by a previous `pfenable(1M)` request. If `–E` is given for the *username*, the privileges will no longer be granted to "all users". Note, that privileges granted to specific users will not be revoked.

This utility may only be invoked by the superuser.

OPTIONS

`–A` revokes *username*'s ability to disable the process affinity mechanism.

`–D` revokes *username*'s ability to use preemption control.

`–c` revokes *username*'s ability to bind processes to particular CPUs.

`–C` revokes *username*'s ability to bind processes exclusively to particular CPUs.

`–p` revokes *username*'s ability to run processes at a fixed priority and to set negative nice values.

`–U` revokes *username*'s ability to create processes which may fork an unlimited number of processes.

SEE ALSO

`launchit(1M)`, `mpcntl(1M)`, `pfenable(1M)`, `pfpriv(1M)`, `pfprivs(1M)`, `nice(2)`, `mpcntl(3X)`.

System Administrator's Reference Manual

pfenable(1M)

NAME

`pfenable -` grant special process flags privileges to users

SYNOPSIS

`/etc/pfenable –[ ADcCpU ][ `*username* `... | –E ]`

DESCRIPTION

WARNING:

The use of some of the options offered by this command may severely disrupt the normal functioning of the system.

`pfenable` grants the special privileges specified by the option list to *username* for the manipulation of process flags normally reserved solely for the superuser. If `–E` is given for the *username*, the privileges will be granted to all users.

`pfenable` only grants a privilege to a user. In order to use the capability granted, a user must invoke `launchit`(1M) or issue a process flags call.

This utility may only be invoked by the superuser.

OPTIONS

`–A` enables *username* to disable the process affinity mechanism.

`–D` enables *username* to use preemption control.

`–c` enables *username* to bind processes to particular CPUs.

`–C` enables *username* to bind processes exclusively to particular CPUs.

`–p` enables *username* to run processes at a fixed priority, and allows *username* to set negative `nice` values.

`–U` enables *username* to create processes which may fork an unlimited number of processes.

SEE ALSO

`launchit`(1M), `mpcntl`(1M), `pfdisable`(1M), `pfpriv`(1M), `pfprivs`(1M), `nice`(2), `mpcntl`(3X).

System Administrator's Reference Manual

pfpriv(1M)

NAME

`pfpriv -` list special process flags privileges granted to users

SYNOPSIS

`/etc/pfpriv [` *username* ... `| –E ]`

DESCRIPTION

`pfpriv` lists the special privileges, granted by a prior `pfenable(1M)` request, to *username* for the manipulation of process flags normally reserved solely for the superuser. If `–E` is given for the *username*, the special privileges that are granted to all users will be listed.

SEE ALSO

`launchit(1M)`, `mpcntl(1M)`, `pfdisable(1M)`, `pfenable(1M)`, `pfprivs(1M)`, `nice(2)`, `mpcntl(3X)`.

System Administrator's Reference Manual

pfprivs(1M)

NAME

`pfprivs` - list all users with special process flags enabled

SYNOPSIS

`pfprivs`

DESCRIPTION

`pfprivs` is similar to `pfpriv(1M)`, except that it lists all users who currently have special `processflags` privileges enabled, along with their respective privileges (`pfpriv` shows privileges for specified users only).

Normally, only the superuser can use `pfprivs`, since it requires read access to `/dev/kmem`.

FILES

`/unix`

`/dev/kmem`

SEE ALSO

`launchit(1M)`, `mpcntl(1M)`, `pfdisable(1M)`, `pfenable(1M)`, `pfpriv(1M)`, `nice(2)`, `mpcntl(3X)`.

System Administrator's Reference Manual

pic(1M)

NAME

`pic` - troff preprocessor for drawing pictures

SYNOPSIS

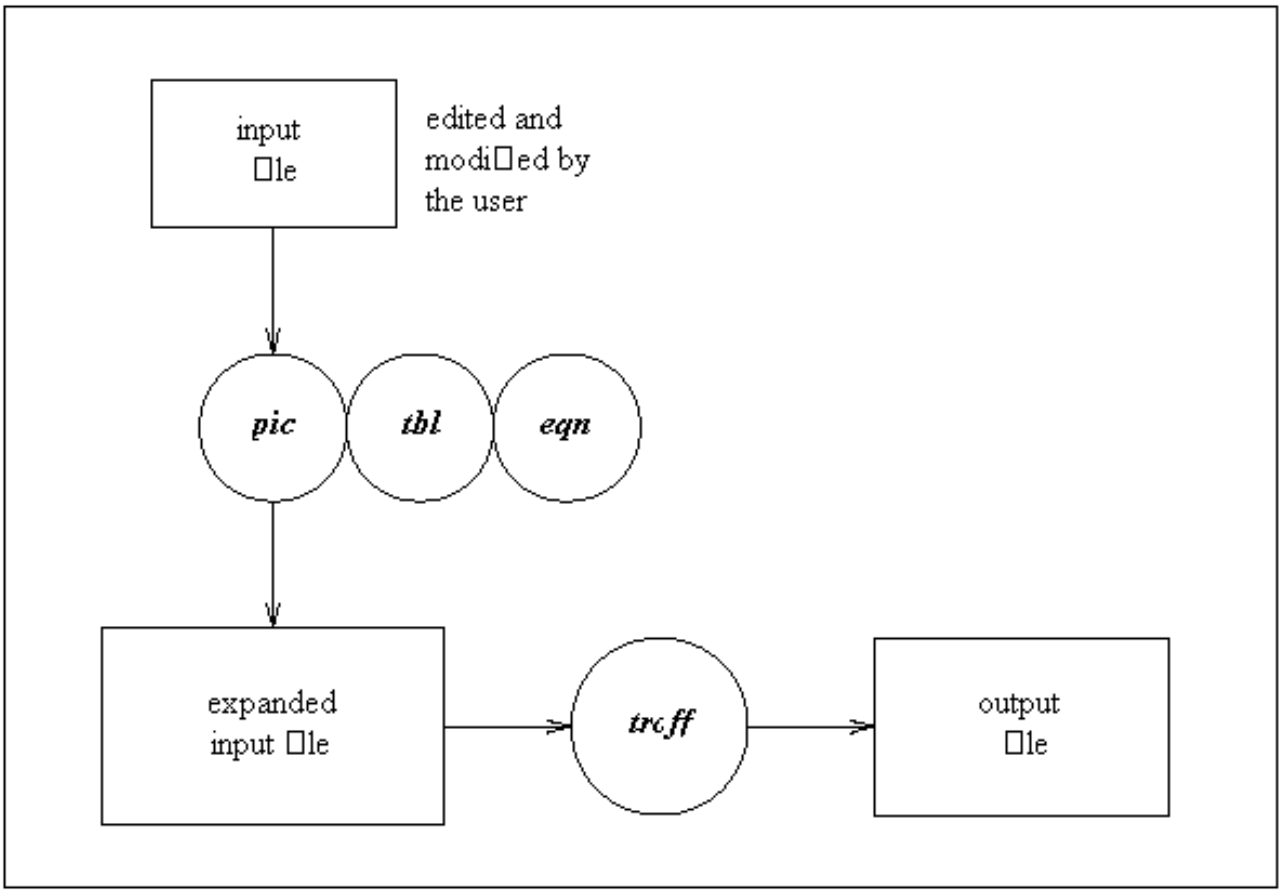
`pic` [`-T` *device*] [`--`] [`-`] [*file*] ...

DESCRIPTION

Requirements

Like the `tbl`(1M) program for tables or the `eqn`(1M) program for mathematical formulae, the `pic` program for generating graphics is a preprocessor which translates the special source code embedded in troff source texts to troff source text. Only then is this expanded, "translated" input text as well as the remaining "normal" input text actually formatted.

The entire context is again illustrated in the following figure.



In contrast to `tbl` and `eqn`/`neqn`, `pic` can only be used in combination with `troff`:

`pic` *option file* ... | `troff` ...

If you use `pic`, `eqn` and `tbl`, the pipe order is as follows:

`pic` *option file* ... | `tbl` | `eqn` | `troff` ...

OPTIONS

The following option is permitted:

–T *device*
: The output is prepared for the output *device*. The complete `/usr/lib/font/dev`*device* directory must be available for *device*.

The following parameters are permitted:

––
: End of list of options. This entry is required if *file* begins with –.

–
: If no file is specified, or if – is specified as an argument, standard input is read.

file
: Name of troff input file.

INTRODUCING PICTURES IN TROFF TEXTS

Picture descriptions are delimited in troff input text by `.PS` and `.PE` (on separate lines). As an option after `.PS`, you can specify the width and height of the picture in inches; the picture is then stretched and dragged to fit this size. Otherwise, standard dimensions or those specified in the picture description are used. If only the width is specified, the height is adjusted proportionally. If the `.PS` line appears as

`.PS <`*file*

it is replaced by the contents of *file*. *file* may contain delimiters, but this is optional.

If you use `.PF` instead of `.PE`, the text position before the start of the picture is restored once the picture has been created. This allows you to output text as well as the picture for example.

`pic` passes through the `.PS`/`.PE` delimiters, and appends the calculated height and width of the picture to `.PS` (before any values that may be present). The delimiters and height and width values of the picture can then be processed again using a macro package. For example, you can center all pictures as follows:

```
.de PS
.ne \\$1u
.in (\\n(.lu-\\$2u)/2u
..
.de PE
.in
..
```

SYNTAX OF PIC INPUT LANGUAGE

The description of a picture basically comprises a number of figures each with specific attributes; these will now be explained. Each combination of figure and attribute (= object or element) appears in a separate line. You can also specify several elements on one line, separating them with a `;` character. If the description of an element is very long, you can split it over several lines, though the last character of each line before a follow-on line must be a backslash, i.e. `\`.

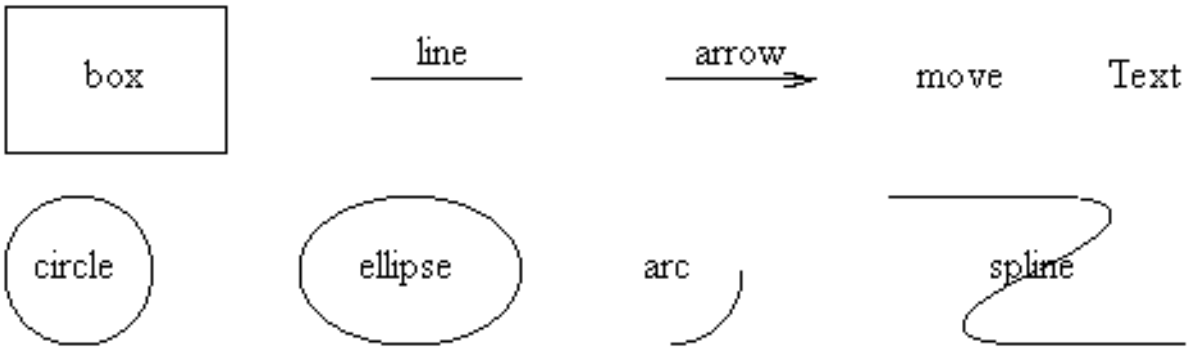
You can also add comments to the description of the picture. These comments are introduced by a # character and terminate at the line end.

You will see later how you may also use variables and set markers, but first let us look at the notation: Variable names begin with a lowercase letter, while marker names begin with an uppercase letter. Both variables and markers retain their values in subsequent pictures in the same document.

Each line beginning with a period is interpreted as a `troff` command and is passed through unchanged. You must not use control characters for vertical spaces or for changing the line spacing. You can, however, change the font and point size.

BASIC FIGURES AND THEIR ATTRIBUTES

The basic figures in `pic` are boxes, lines and arrows, circles, ellipses, arcs, splines, text in quotation marks as well as moves. You can see all of these in the diagram below:



An `arrow` is another word for a `line ->`.

Each of these basic figures can be assigned attributes which basically influence the size, positioning and labeling of the figures. You can also make each figure invisible by appending the word `invis(ible)`. This makes it easier, for example, to position picture labels. Each attribute comprises a keyword possibly followed by a value. For example:

`height 3.5` or `ht 3.5`

means that the previous basic figure will have a height of 3.5 inches.

Default values can be used to replace any missing attributes and values. Not every attribute can be applied to every basic object, and unsuitable attributes are simply ignored. The possible **attributes** for each figure are listed below:

Figure
Permitted Attributes

`box` h(eigh)t, wid(th), at, same, dotted, dashed, invis, "*Text*"

`circle`, `ellipse`
chop, rad(ius), diam(eter), h(eigh)t, wid(th), at, same, invis, "*Text*"

`line`, `arrow`
up, down, left, right, h(eigh)t, wid(th), from, to, by, then, at, same, dotted, dashed, invis, <-, ->, <->, "*Text*"

`arc` up, down, left, right, h(eigh)t, wid(th), from, to, at, rad(ius), invis, cw (= clockwise), <-, ->, <->, "*Text*"

`spline`
up, down, left, right, h(eigh)t, wid(th), from, to, by, then, at, same, invis, <-, ->, <->, "*Text*"

`move` up, down, left, right, to, by, same, "*Text*"

"*Text*"
at, by

Box

The keyword `box` creates a standard sized box (0.75 inches wide, 0.5 inches high) at the current position:

A number of figures are directly concatenated without any spaces in between, i.e.

`box; box; box`

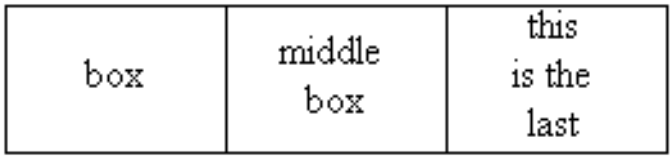
produces

You could also have entered:

```
box□
box□
box
```

To label the boxes, you append text attributes to the names of the figures in double quotes:

```
box "box"□
box "middle" "box"□
box "this" "is the" "last"
```

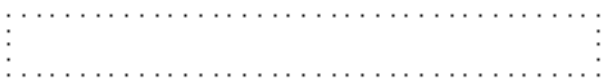


Quotation marks must be inserted even if the text does not contain spaces. Each text attribute is output in its current point size and type and centered horizontally and vertically; if there are several text attributes for the same figure they are output one above the other assuming the current line spacing.

Dashed or dotted boxes are an interesting variation. In this case, you enter `dashed` or `dotted` after `box`.

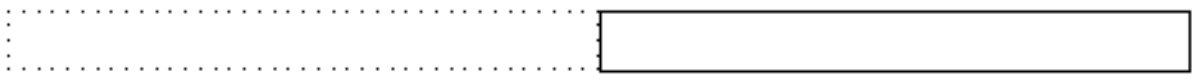
You can also make the box any size you wish. Thus, for example, the following box is produced with

```
box dotted wid 2i height 0.2
```



Dimensions are always interpreted as inches. The letter "i" can be specified, but this is optional. The dimensions are only valid for the basic figure specified immediately beforehand. Even if a basic figure of the same type follows, the default size applies again initially. You can retain the dimensions, however, by specifying the keyword `same`:

```
box dotted wid 2i height 0.2; box same
```



You can also specify dimensions after `dotted` and `dashed`. These dimensions are explained in the section "Lines and arrows".

Move

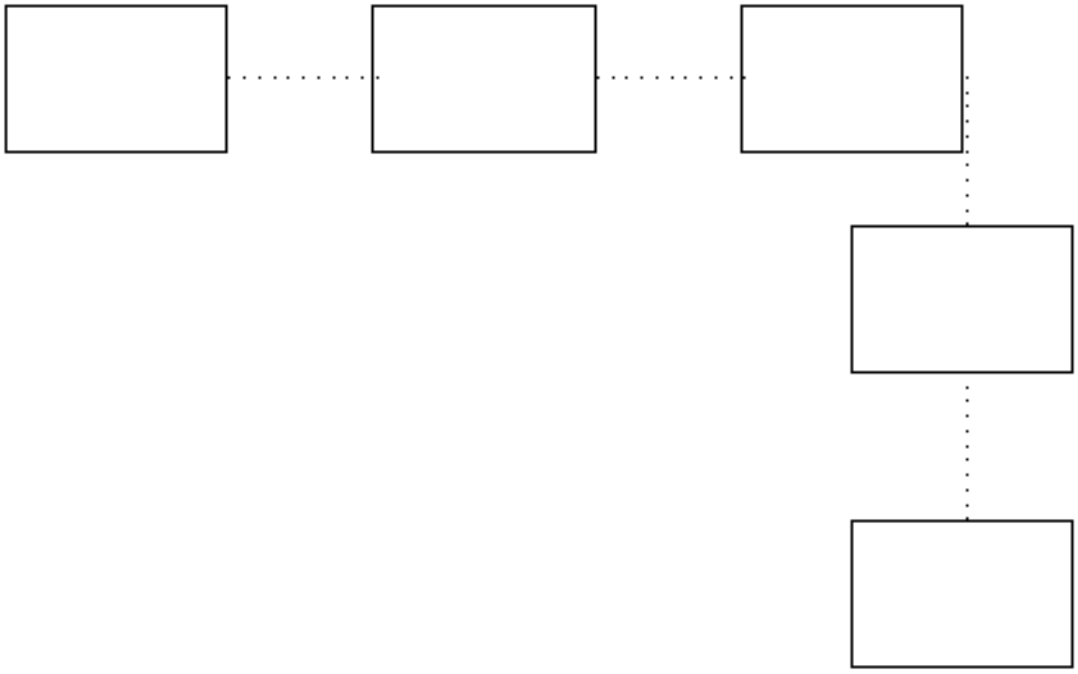
Basic figures are generally output in direct succession from left to right. If you want space in between, you should use `move`:

```
box; move; box; move; box
```

Since `move` is not a figure, you use `up`, `down`, `left` or `right` and possibly a value to specify the size of the

move and not "width" or "height". But be careful: Each of these attributes also changes the general direction of the basic figures. Thus, for example, the following picture is produced with:

```
box; move; box; move; box; move down; box
move; box;
```



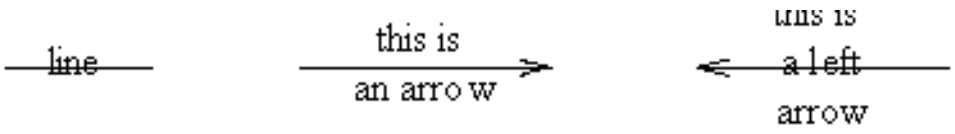
(The `move`s are indicated by dotted lines.)

Lines and arrows

Lines and arrows are produced with `line` and `arrow`. `arrow` is not a basic figure, rather a shortcut for writing `line ->` and produces a line with an arrow at the end. You could also use arrowheads at the beginning or at both ends using `line <-` or `line <->`.

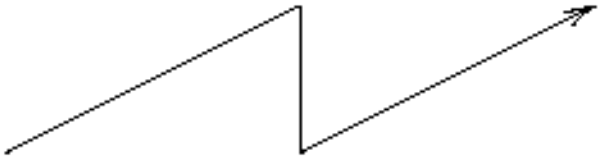
Take a look at the text output at the lines and arrows in the following examples:

```
line "line"; move; arrow "this is" "an arrow"
move; line <- "this is " "a left " "arrow"
```



If lines and arrows are to have different lengths and directions, you specify the direction of movement and spacing from the current position as with `move`, i.e. `up`, `down`, `left` and `right` and possibly a distance. For example:

```
line up .5 right 1
line down .5
arrow up .5 right 1
```

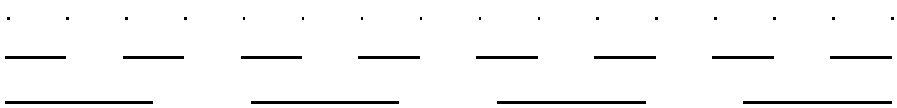


If you do not specify a distance after `up`, `down` etc., `pic` will use default values. However, you cannot specify a distance without a direction. Once more beware: In this case also, a direction attribute changes the general direction of the picture.

If you specify a "width" or "height" for `line` or `arrow`, this only refers to the dimension of the arrowhead. These attributes are meaningless for `line`s without `->`, `<-` or `<->`. As with boxes, lines can be dotted or dashed:

```
line dotted; move; line dashed
```


Both attributes can be assigned dimensions. A number after `dotted` defines the distance between the dots. The dimension for `dashed` defines both the width of the dashes as well as the space in between. For example:



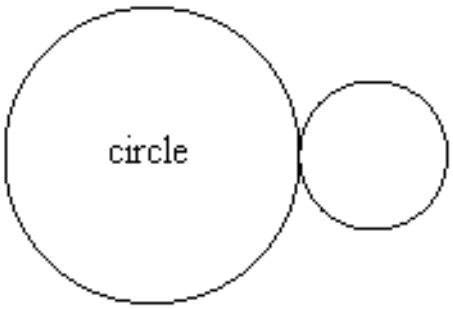
is produced from the following input (separate pictures):

```
line right 3i dotted 0.2
line right 3i dashed 0.2
line right 3i dashed 0.5
```

Circle

Circles are produced using the word `circle`. The size of the circle can be specified as a radius ("rad") or a diameter ("diam" or "wid") if the default is not to be used:

```
circle rad 0.5 "circle"; circle diam 0.5
```

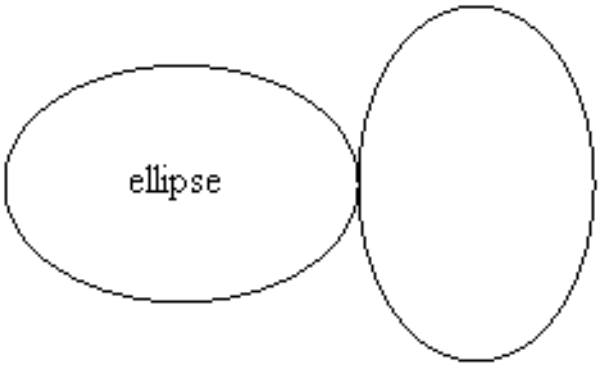


Circles and arcs are never stretched or dragged if a dimension is specified after `.PS`, rather they are increased or reduced proportionally. Circles and arcs cannot be dotted or dashed.

Ellipse

Ellipses are produced using the word `ellipse`. The size of an ellipse is specified with height ("ht") or width ("wid" or "diam") if the default is not to be used.

```
ellipse ht 0.8 wid 1.2 "ellipse"
ellipse wid 0.8 ht 1.2
```

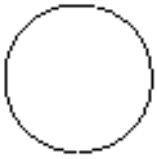


The radius, if specified ("rad"), only affects the width of an ellipse.

Arc

Arcs are produced with `arc`. Default values rotate the arc 90° in an anticlockwise direction. Thus, for example,

```
arc; arc; arc; arc
```



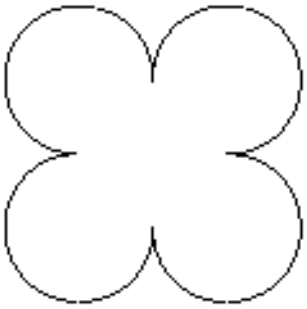
naturally produces a circle. The direction can also be reversed with the attribute `cw` (= clockwise):

```
arc cw; arc cw; arc
```

The default radius is the same as for circles but can be changed with the `rad` attribute.

Each arc changes the general direction of the picture in the direction of its last segment. Take a look at the example below:

```
arc; arc; arc; move
arc; arc; arc; move
arc; arc; arc; move
arc; arc; arc; move
```



If you have established the sequence in which the arcs in this picture are drawn, then you have understood

the general principle.

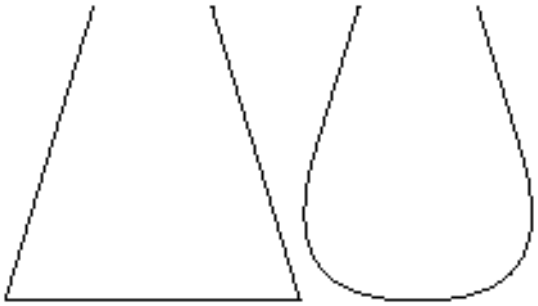
If an arc is to start in a different direction than the current one, specify `up`, `down`, `left` or `right` as an attribute (without a distance). If arcs are to have completely different lengths and directions, you should use the attributes `from` and `to`, specifying a position in each case (see section "Coordinates").

You use the attributes `<-`, `->` or `<->` for arrowheads in arcs; `height` and `width` only affect the arrowhead in arcs. Arcs cannot be dotted or dashed.

Spline

A spline is a rounded curve drawn along a series of straight lines. The syntax for a spline is the same as the syntax for a series of linked straight lines (polygons), though using the keyword `spline`. A polygon and spline are shown side by side in the picture below:

```
  line left .3 down 1 then right 1 then \
  left .3 up 1
  move right
spline left .3 down 1 then right 1 then \
  left .3 up 1
```



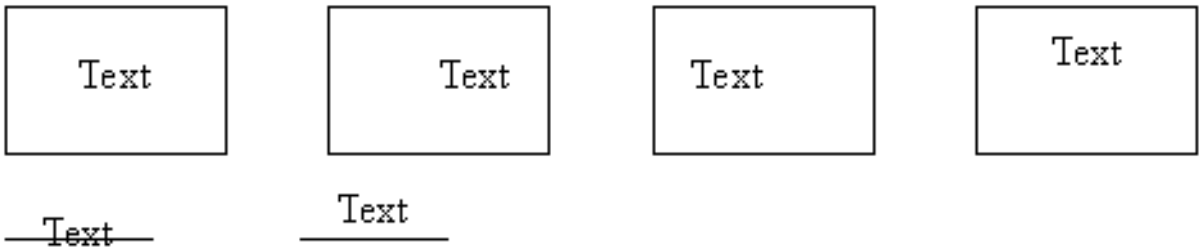
The elements in a polygon or spline are specified as a series of vertices separated by "then". The vertices are generally specified as coordinates (see section "Coordinates").

Arrowheads can be output at the tips of the polygon or spline, where `height` and `width` only affect the arrowheads. Splines cannot be dotted or dashed.

Text

Text attributes are enclosed in double quotes. A text attribute is generally output at the center of the basic figure (text position). However, there are also additional attributes for text attributes, which allow the text to be output left or right justified from the current position or half a line higher or lower. These additional attributes are `ljust`, `rjust`, `above` and `below`. They must not be combined however.

```
box "Text"; move
box "Text" ljust; move
box "Text" rjust; move
box "Text" above; move
line "Text"; move
line "Text" above
```



If several text attributes are specified for a basic figure, they are output one above the other to the center of

the figure, unless specified otherwise. An additional attribute only affects the directly preceding text attribute.

Text is generally an attribute of another object, but can also be created independently:

```
box; move right 3; "Text"
```

Stand-alone text can also be assigned additional attributes.

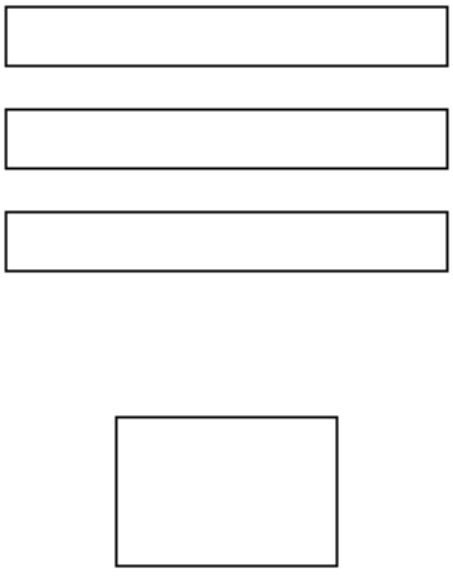
DIMENSIONS OF BASIC FIGURES

Each basic figure (`box`, `circle` etc.) has default measurements such as height, width, radius etc.:

Basic figure	Dimension
`box`	0.75 inches wide × 0.5 inches high
`move`	0.5 inches vertically and horizontally
`line`/`arrow`	0.5 inches vertically and horizontally
`circle`	0.5-inch diameter = 0.25-inch radius
`ellipse`	0.75 inches wide × 0.5 inches high
`arc`	0.25-inch radius
`spline`	0.5 inches
"*Text*"	0 inches wide and high

The description of the basic figures explains how to changes these dimensions for individual basic figures using attributes. If you specify an attribute like `width`, then only one object is changed. If a figure is to be the same size as a previous object of the same type, append the name `same`. For example:

```
down; box ht 0.2i wid 1.5i; move down 0.15i
box same; move same; box same; move; box
```



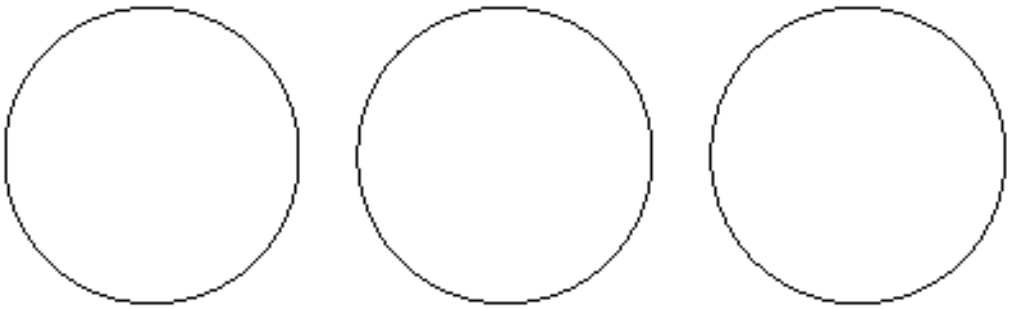
By assigning new values to particular variables, you can permanently change the default dimensions for the

basic figures. These names are as follows (see section "Variables and arithmetic" below):

```
boxwid, boxht
movewid, moveht
arrowwid, arrowht
linewid, lineht
dashwidcircleradarcradellipsewid, ellipseht
textwid, textht
```

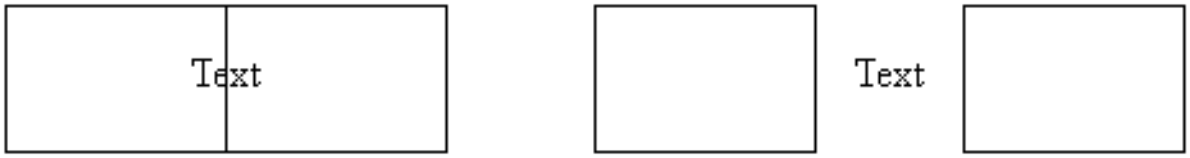
For example, if all circles are to be large and be placed closely together, you can assume the following general setting:

```
circlerad = .5; movewid = 0.2i
circle; move; circle; move; circle
```



Although text is always output as normal, the space taken up by the text is 0 by default (`textwid` = `textht` = 0). Take a look at the following example:

```
box; "Text"; box;
move; textwid = .5
box; "Text"; box;
```

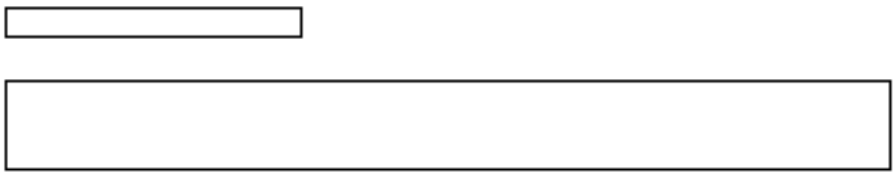


`pic` generally works with inches as a unit of measurement, but offers the option with the default variable `scale` of using a different unit within the description of the picture – e.g. centimeters. If you set `scale` to a value ≠ 1 all measurements will be divided on the basis of this value. `scale=2.54` therefore causes all measurements to be interpreted as centimeters.

The number specified as the width in the `.PS` line overrides the measurements defined in the picture. This is how you set a picture to a particular size. For example:

```
boxwid = 1; boxht = .1; box
```

If you enter this information with `.PS` without parameters or with `.PS 3`, the following is output:



POSITIONING

Coordinates

`pic` figures are generally output in direct succession from left to right below the last formatted text. However,

objects can be placed at any position. `pic` uses a typical two-dimensional coordinate system in which every point has an x and a y coordinate. The position 0,0 is always the starting point for the first object in a picture. The x,y position of a box, a circle or an ellipse is its geometrical centerpoint. The position of a straight line or move is the starting point. The position of an arc is the centerpoint of the circle.

You can change the absolute positioning of a figure with the attribute "at" and a pair of coordinates or the relative positioning with the attribute "by" and a pair of coordinates. In the case of moves, lines, arrows and arcs, use "from" and "to" (or simply `to`) with coordinates to specify the positioning.

If you omit the attribute `from`, the move, line or arc begins at the position at which you are now located and runs to the point specified by `to`.

The x and y coordinates can also be changed individually using the familiar attributes `left`/`right` and `up`/`down` and a relative value.

A position is therefore a pair of x,y coordinates, but can also be specified in another way. The *expression , expression* pair of coordinates can be enclosed in parentheses.

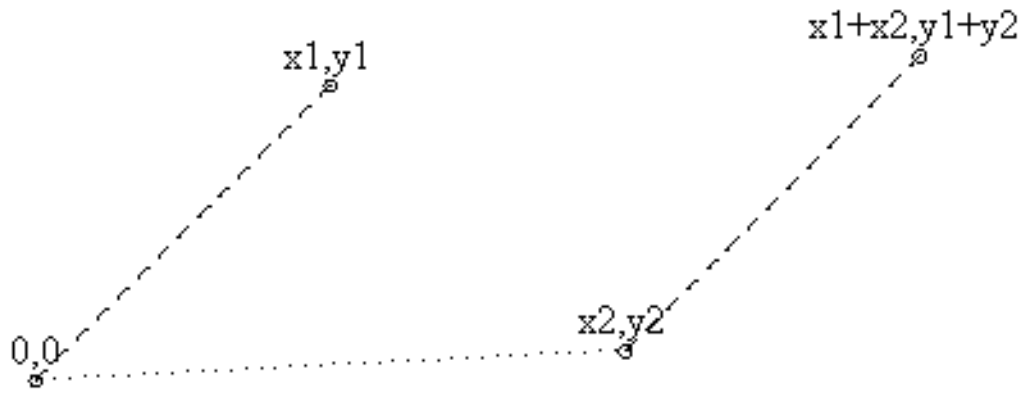
The size and position attributes such as `ht`, `wid` and `at` can be written in any sequence. As a result,

```
circle rad .3 at 1.5,.8
circle at 1.5,.8 rad .3
```

have the same effect.

Positions can be added, subtracted and projected in order to create new positions:

$(x_1,y_1)+(x_2,y_2) = (x_1+x_2,y_1+y_2)$

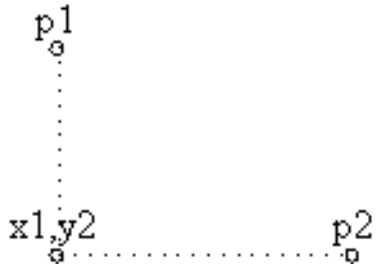


If p_1 and p_2 are positions with the coordinates (x_1,y_1) and (x_2,y_2). Then

(p_1,p_2)

references the projection of the two positions

$(p_1.x,p_2.y)$ $(= (x_1,y_2))$

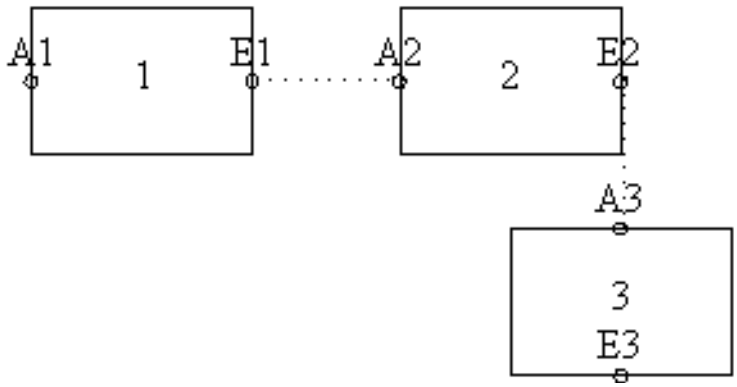


Start and end points for figures

Objects are linked in the sequence defined by the last `up`, `down`, `left` or `right`. The starting point for the

second object is therefore the end point for the first object. The start and end points for boxes, circles and ellipses are always at the center of the opposite side of the figure in the current direction, and represent the beginning and end of lines, splines, moves and arcs. For example:

```
box "1"; move; box "2"; move down; box "3"
```

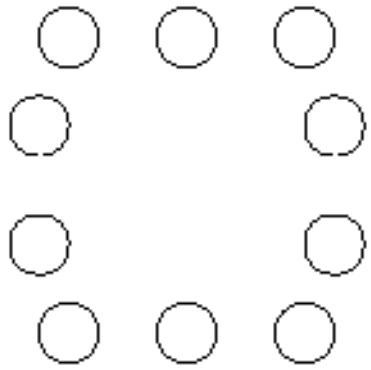


The start and end points for boxes are indicated with A*i* and E*i*. The end point of a figure is also the starting point of the next figure, i.e. E2 is the starting point of the next `move` (`move down`) and A3 is its end point.

Direction

The initial direction set for every picture is `right`, i.e. objects are linked from left to right. If you specify a direction as its own object or as an attribute, all subsequent objects are linked in this direction. For example:

```
circlerad = .1; movewid = 0.2i; moveht = .2
circle; move; circle; move; circle; move down
circle; move; circle; move; left
circle; move; circle; move; circle; move up
circle; move; circle; move
```



Every other picture (`.PS-.PE`) again runs from left to right.

Markers

Objects can be marked and named to allow their position to be addressed again later. For example:

```
Cir1:
   circle; move; circle; move; circle; move
   circle; move to Cir1; "circle 1"
```

The marker in this example is Cir1. Marker names must begin with an uppercase letter. The name indicates the position of the next figure. In most cases, this position is the geometrical centerpoint. You can use markers as follows:

```
line from Cir1 to Cir2
move to Cir1 down .5 right 0.3
line to Cir1 + 0.5,0.3
circle at Cir1 + 0.5,0.3
```

In the last two examples, *marker* + x,y creates a new position. The reserved name `Here` identifies the current position:

```
Position1: Here
```

Markers are variable and can also be changed within a picture, and even redefined as follows:

```
Position1: Position1 + 1,1
```

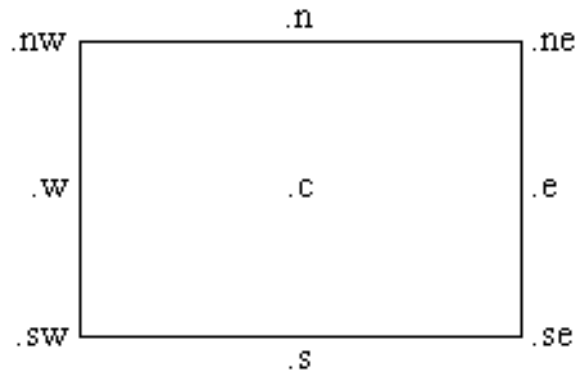
Instead of a suitable marker, you can also refer to previously described objects of any type using the word `last`. In this picture description,

```
box "1"; circle "2"; box "3"
```

`last box` refers to box 3, `last circle` to circle 2 and `2nd last box` to box 1. Objects of the same type are numbered consecutively in each picture. Boxes 1 and 3 are thus `1st box` and `2nd box` (`1th` is also permitted as is `1st` or `3rd`).

Corners

So that not only **the** position of an object can be addressed, but also a number of its boundary points, most objects have "corners" named in accordance with compass points:

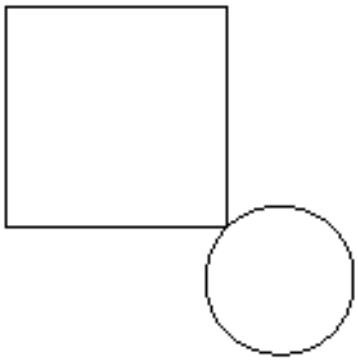


The descriptors `.r`, `.b`, `.l` and `.t` (for "right", "bottom", "left" and "top") can also be specified for the corners `.e`, `.s`, `.w` and `.n`. Like corners, lines and arrows have a `.start` and `.end`. Arcs only have `.c`, `.start` and `.end`. Apart from this type, almost every combination of `left`, `right`, `top`, `bottom`, `upper`, `lower` and `center` is possible for describing particular points in a basic figure. Thus `last box.ne` corresponds to `upper right of last box`.

Objects can be positioned by allowing a corner of one object simultaneously form the corner of another object. The `with` attribute allows this type of positioning. For example:

```
box ht 0.75i wid 0.75i
circle with .nw at last box.se
```

produces:



Finally, you can position along a line defined by two points:

fraction [`of the way`] `between` *position*$_1$ `and` *position*$_2$

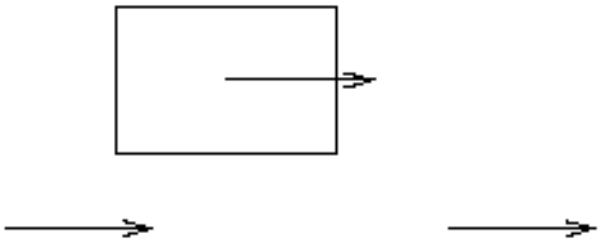
This notation can be shortened as follows:

fraction `<`*position*$_1$`,` *position*$_2$`>`

The value specified by *fraction* can be greater than 1 or less than 0. For example:

```
box
arrow right from -1/2 <last box.sw, last box.ne>
arrow right from 1/2 of the way between \
last box.nw and last box.se
arrow right from 3/2 <last box.nw, last box.se>
```

produces:

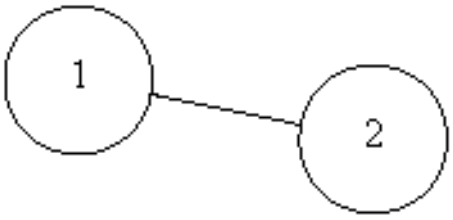


Concealed lines

There are many cases where a line is to intersect a circle at a point, which is not one of the eight compass points allowed for in `pic`. A line is drawn to the centerpoint and a section of the line is truncated using the `chop` attribute:

```
circle "1"
circle "2" at 1st circle + 1,-.2
line from 1st circle to 2nd circle chop
```

produces:



`chop` generally cuts a line at each end by the `circlerad`, whose size can be changed as follows:

`line ... chop` *length*

reduces both ends by *length*,

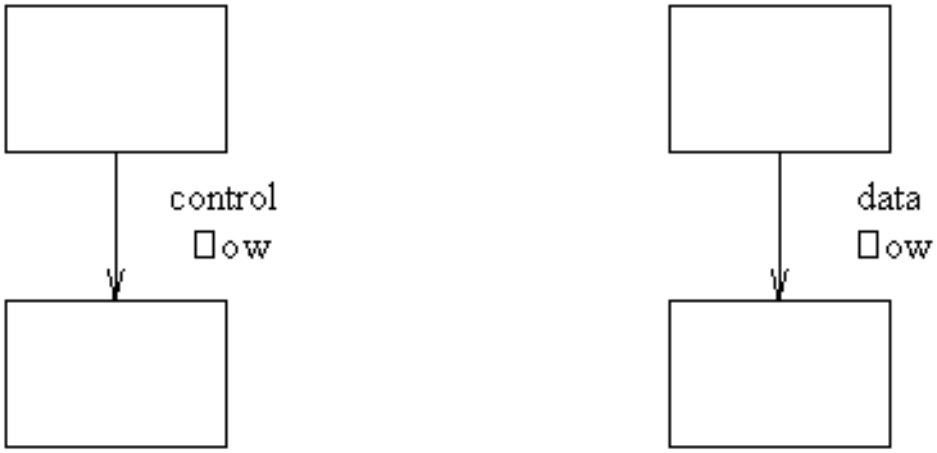
`line ... chop` *length*$_1$ `chop` *length*$_2$

reduces the start by *length*$_1$ and the end by *length*$_2$.

Blocks

A block is formed by a sequence of `pic` statements enclosed in parentheses. A block counts as a single object and can be manipulated like an ordinary box which contains all of the figures in the block. For example:

```
[down; box; arrow; box]
move; move; move
[down; box; arrow; box]
"control" "flow" at 1st [].e
"data" "flow" at last [].e
```



A block is treated like a unit, but has its own direction of movement and its internal objects cannot be addressed: `last box` is not known outside of the blocks, but you can use `last []` to address the block as a whole.

Blocks have compass points like boxes because every block is delimited by an invisible box. A block can also be positioned by either moving it in its entirety or using an internal marker:

```
[ ...; Cir1: ...; ... ] with .Cir1 at ...
```

Blocks are linked with the following figure in the same way as boxes (i.e. at the endpoint of the adjacent box).

Names of variables and markers within a block are only valid within the block; they do not come into conflict with external variables and markers of the same name. However, internal markers can also be used outside the block in the following ways:

`last [].Cir1` or `Block1.Cir1`

if Block1 is a marker, which is assigned to the block:

`Block1: [ ... ; Cir1: ...; ]`

Even though blocks may be nested, the nesting of markers such as Block1.Cir1 can only be one level deep. Cir1 can be described more explicitly however. For example `Block1.Cir1.e` or `start of Block1.Cir1`.

Command sequences appear in curly brackets {...} alongside the actual blocks. These commands cause the current position and direction of movement to be exactly the same following the closing bracket as for the open bracket. This is their only effect.

VARIABLES AND ARITHMETIC

Wherever you are to use values – thus, for example, after `down` or `wid` –, you can also define variables or any

other expressions. As with programming, absolute coordinates are time consuming and prone to error if changes need to be made subsequently. Variables allow you to specify parameters for your picture. For example:

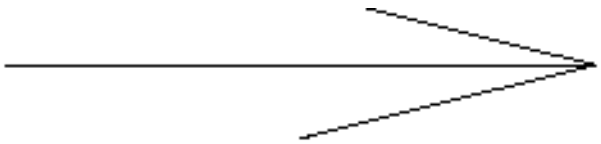
```
g = .75; h = 1.2
box wid g/3 ht h*1.5
move to last box - (a, b*2)
```

Expressions can use the operators +, –, *, / and %, with round brackets used for summaries. Please note that variable names should begin with a lowercase letter.

There are already a number of predefined variables which establish the default dimension of the basic figures. You can changes their values in a picture at any time, and they are preserved until they are defined again in the same document. The predefined variables and their initializations are as follows:

Variable	Default Value	
arcrad	0.25 i = 0.635 c	Radius of arc
arrowwid	0.05 i = 0.127 c	Width of arrowhead
arrowht	0.1i = 0.254 c	Length of arrowhead
boxwid	0.75 i = 1.905 c	Width of box
boxht	0.5i = 1.27c	Height of box
circlerad	0.25 i = 0.635 c	Radius of circle
dashwid	0.05 i = 0.127 c	Width of dash and space in a dashed line
ellipsewid	0.75 i = 1.905 c	Width of ellipse
ellipseht	0.5i = 1.27c	Height of ellipse
linewid	0.5i = 1.27c	Length of horizontal line
lineht	0.5i = 1.27c	Length of vertical line
movewid	0.5i = 1.27c	Distance of horizontal motion
moveht	0.5i = 1.27c	Distance of vertical motion
scale	1i = 2.54c	Scale dimensions
textwid	0i	Width of space required by a text as an object
textht	0i	Height of space required by a text as an object

For example, you can enlarge an arrowhead by changing the following predefined variables as shown below:



```
arrowwid = .5
```

```
arrowht = 1
linewid = 2
arrow
```

You can use the default variables as required in expressions. Other possible values are other variables, height, width, radius and x and y coordinates of an object or a corner. For example:

```
spacing       # Variable
Cir1.x        # x-coordinate of Cir1
Cir1.ne.y     # y-coordinate of northeast corner of Cir1
Cir1.diam     # Diameter of Cir1
2nd last circle.rad # Radius of second last circle
```

Expressions are calculated as broken values. All digits describing measurements are interpreted as inches, which are `scaled` when output.

PROGRAMMING OPTIONS FOR PIC

Apart from the use of variables and the creation of expressions as we are now familiar with, the options for programming in `pic` are not very extensive, but are ideal for creating pictures. In addition to the most important elements like definitions and macros, the loop construction as well as their conditional execution, there are other elements such as formatting of values, importing other files as well as a number of integrated mathematical functions.

Definitions and macros

`pic` provides a macro facility, which in its simplest form is identical to that in `eqn`:

`define` *name* `'`*replacement_text*`'`

Every time *name* appears in the following pictures, it is replaced by *replacement_text*. The separator (`'` in this case) can be any character that does not appear in the *replacement_text*. The *replacement_text* can include line feeds.

Macros can also be defined with arguments. The characters $1 to $9 can appear in the replacement string of a macro definition and are replaced by the corresponding actual arguments when the macro is invoked. To invoke a macro with arguments:

name`(`*arg1*`, `*arg2*`, ...)`

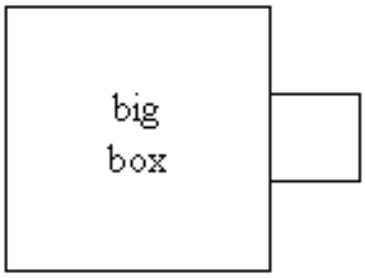
Non-existent arguments are replaced by null strings. Pairs of coordinates may be enclosed in parentheses (as in (x,y)), so that they can be included in a macro argument. For example:

```
define square Xbox ht $1 wid $1 $2X
```

The input

```
   square(.9, "big" "box"); square(.3)
```

produces a large labeled and a small unlabeled box:



Loops

Loops for repeated execution of processes of a similar type are written as follows:

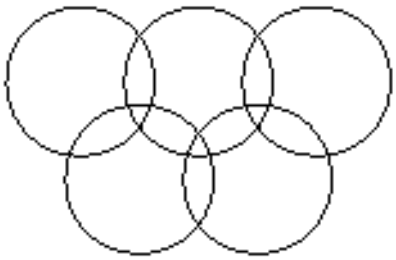
`for` *variable* `=` *start_value* `to` *end_value* [`by` *increment*] `do`
`X`

```
 ...    # loop□
 ...    # body□
X
```

X is any character that does not appear in the loop body. 1 is used if no *increment* is specified. For example:

```
for i = 1 to 5 do□
X□
    circle at (i*0.2,0+0.33*(i%2))□
X
```

produces: □



Conditional execution

The syntax for conditional execution of `pic` statements is as follows:

```
if expression then X□
 ...□
X [else X□
 ...□
X]
```

X is any character that does not appear in the enclosed `pic` statements. Depending on whether *expression* is true (≠ 0) or false (= 0), the segment after `then` or `else` is executed. If the `else` branch is missing and *expression* is false, nothing will be executed. The true-false semantic thus relates to the C language, and not `nroff`/`troff`.

In addition to the arithmetic operators, conditional and logical operators for C are also permitted:

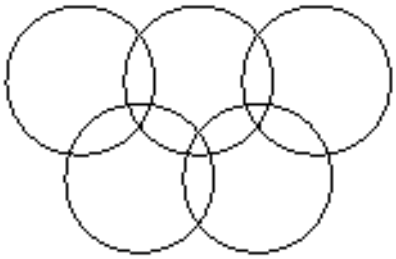
	Operator	Meaning
conditional operators	==	equal to
	!=	not equal to
	<	less than
	<=	less than or equal to
	>	greater than
	>=	greater than or equal to
logical operators	!	(logical) Not
	&&	(logical) And
	||	(logical) Or

You can even compare strings with the conditional operators `==` and `!=` (more extensively than in C). For example:

```
for i = 0 to 0.8 by 0.2 do□
X□
    if i == 0.2 || i == 0.6 then #□
    circle at (i, 0)□
```

```
    # else #
    circle at (i, 0.33)
    #
X
```

produces:



Formatted values

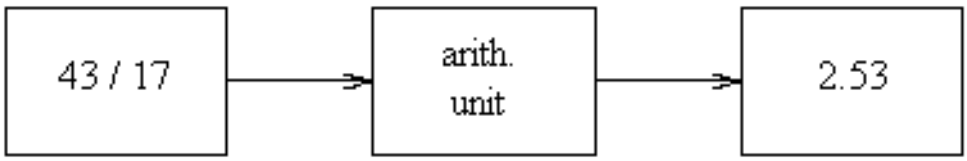
You use the `plot` function to output values calculated in expressions. The formatting that can be used here is similar to `printf()` in C:

`plot` *expression* [*format*] *attributes*

Example:

```
box "43 / 17"; arrow; box "arith." "unit"
arrow; box; plot 43 / 17 "%4.2f" at last box
```

produces:



Integrated functions

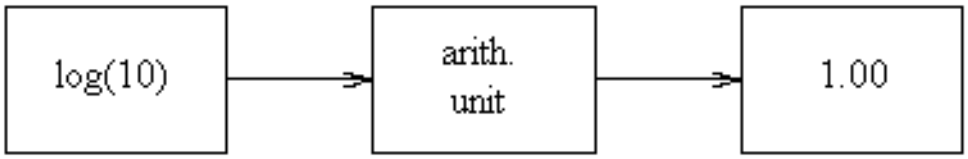
The following mathematical functions can be used to create values in expressions:

`sin(`*expression*`)`	sine of *expression*
`cos(`*expression*`)`	cosine of *expression*
`atan2(`$expression_1$`,` $expression_2$`)`	arctangent of ($expression_1$/$expression_2$)
`log(`*expression*`)`	base 10 logarithm of *expression*
`exp(`*expression*`)`	10 "to the power of" *expression*
`sqrt(`*expression*`)`	square root of *expression*
`int(`*expression*`)`	integer *expression*
`max(`$expression_1$`,` $expression_2$`)`	maximum value of $expression_1$, $expression_2$
`min(`$expression_1$`,` $expression_2$`)`	minimum value of $expression_1$, $expression_2$
`rand(`*expression*`)`	integer random number in the range [1, *expression*]

Examples:

```
box "log(10)"; arrow; box "arith." "unit"
arrow; box; plot log(10) "%4.2f" at last box
```

produces:




```
for i = 0 to 6.3 by 0.1 do□
X□
    "." at (i, sin(i))□
X
```

produces:

Shell commands and linking in files

You can also invoke shell commands from `pic` with:

`sh X` *shell_command* `X`

`X` is again a character that does not appear in the enclosed segment. Please note that like `pic` output, any output of the shell command is sent to standard output.

Apart from the rarely used option of linking in files using `.PS`□`<`*file*, the much more flexible `copy` command can also be used, and can be specified at any position within a picture. The simplest form is:

`copy "`*file*`"`

This statement is simply replaced by the contents of *file*. If *file* contains `.PS`/`.PE`, they are ignored. This means that existing pictures can easily be used in other pictures. The `copy` function also offers the possibility of specifying parameters for macros externally:

`copy "`*file*`" thru` *macroname* or □
`copy "`*file*`" thru X` *replacement_string* `X`

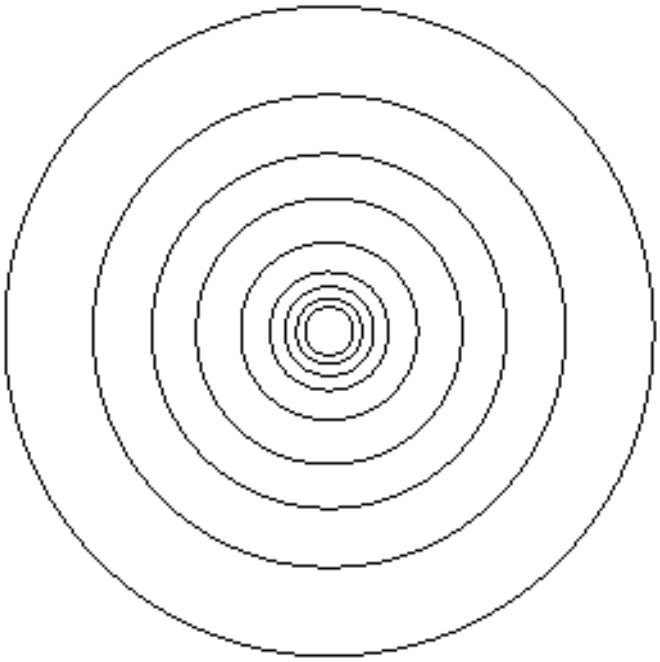
Every line in the copied file invokes the macros specified in either of these ways. This means that the fields of the file content form the macro parameters. For example, both the input

```
define CI X circle rad $1 at 0,0 X□
copy "radian" thru CI
```

and

```
copy "radian" thru X circle rad $1 at 0,0 X
```

create this picture with the file `radian`:



`radian`: output:

```
1.1□
0.8□
0.6□
0.45□
0.3□
0.2□
0.15□
0.11□
0.08
□
```

ERROR HANDLING AND AVOIDING ERRORS

The most harmless `pic` message is

```
pic: xx X yy picture shrunk to aa X bb
```

It means that a picture has exceeded the page margins and has been reduced automatically by `pic`.

If you make a grammatical error when describing a picture, `pic` tries to specify the position at which the error has occurred. For example, the invalid input

```
circle move circle
```

produces the message:

```
pic: syntax error near line n, file abc   context is□
            circle >>> move <<< circle
```

The `>>> <<<` characters mark the position at which the error was established. In some cases, the cause of the error may be before this position. `pic` ends processing following such a message.

`pic` provides the `print` statement for testing and debugging. `print` can output the values in expressions (and also the variables therefore), positions and complete strings to standard error output. For example:

```
box□
print "The current position is"□
print Here□
print "The box width is"□
```

```
print boxwid
```

outputs the following messages to standard error output:

```
The current position is
0.75, 0
The box width is
0.75
```

To check whether `pic` runs without an error message, enter the following command:

```
pic file >/dev/null
```

`pic` therefore does not produce `troff` output, but the error messages and `print` output are shown.

A good way to suppress errors caused by the effects of one picture on another picture in the same document is to place each picture in its own block:

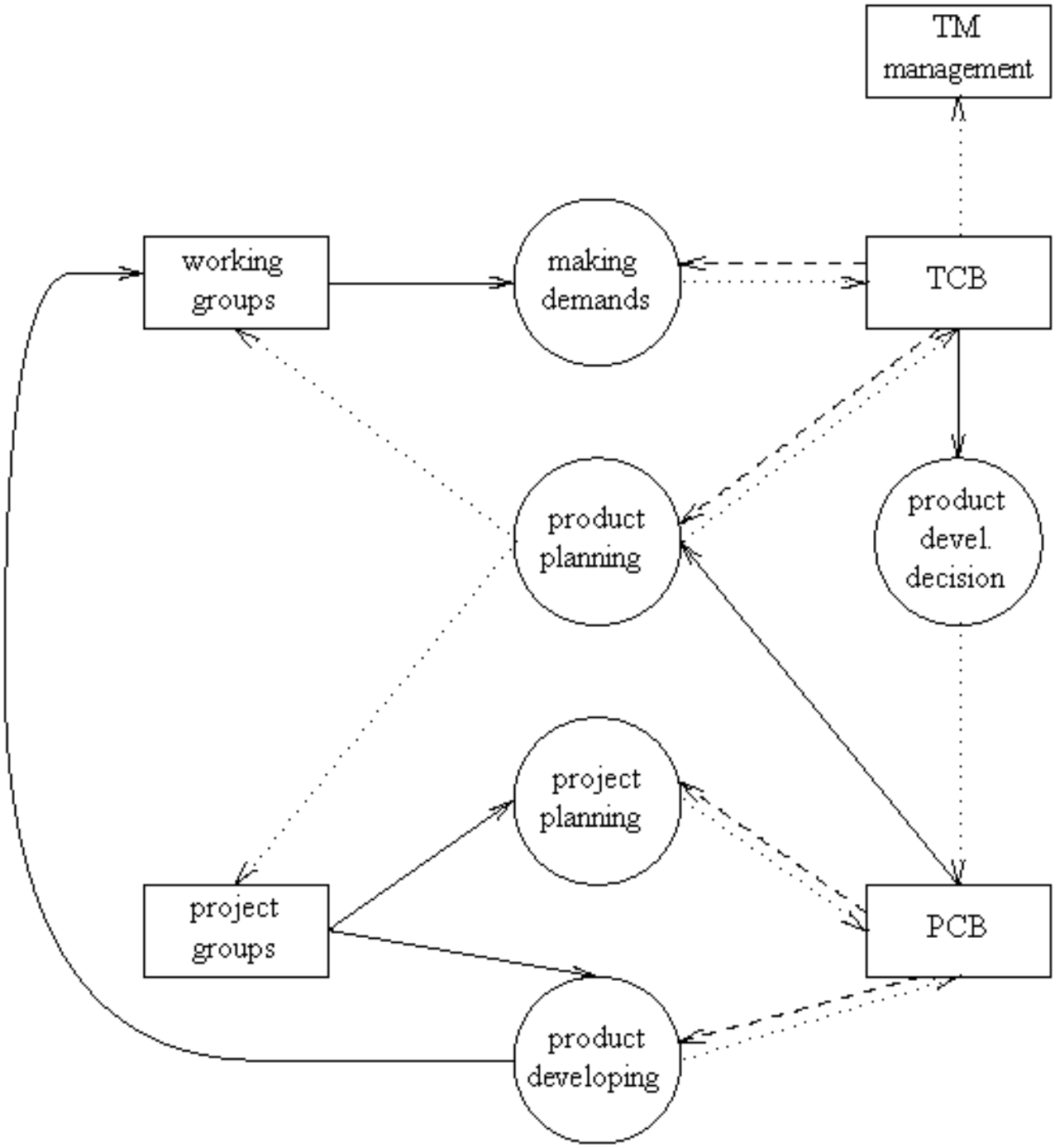
```
 .PS
[ ...
 ...
 ... ]
 .PE
```

This means, for example, that changes to standard variables only affect the current picture.

EXAMPLES

Example 1

Information flow diagram:



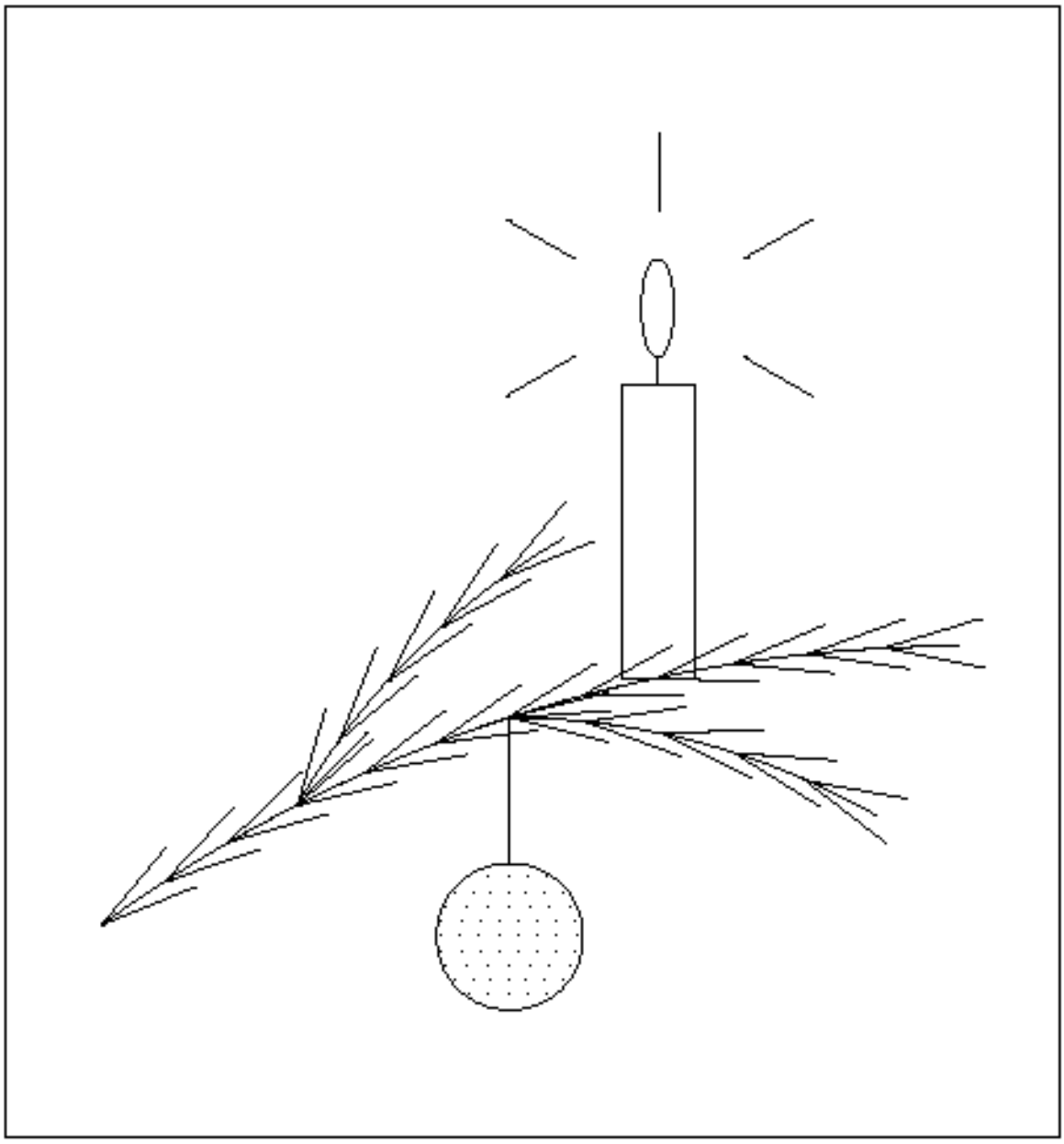
The information flow diagram was produced from the following input:

```
 .PS
scale=2.54
# boxes right
boxht = 1; boxwid = 2
Box_TM: box "TM" "\s-1management\s0"
Box_TCB: box with .n at Box_TM.s -(0,1.5) "TCB"
Box_PCB: box with .n at Box_TCB.s -(0,6) "PCB"
# boxes left
```

```
Box_WG: box with .e at Box_TCB.w -(5.8,0) "working" "groups"
Box_PG: box with .n at Box_WG.s -(0,6) "project" "groups"
# central circles
Circle_demand: circle diam 1.8 with .e at Box_TCB.w -(2,0) \
"making" "demands"
Circle_product: circle rad same with .n at Circle_demand.s -(0,1) \
"product" "planning"
Circle_project: circle rad same with .n at Circle_product.s -(0,1) \
"project"
"planning"
Circle_develop:
circle rad same with .n at Circle_project.s -(0,1) \
"product" "developing"
# right circle
Circle_decision: circle rad same with .w at Circle_product.e +(2.1,0) \
"product" "devel." "decision"
# connecting arrows
arrow from Box_TCB.s to Circle_decision.n
arrow from Box_WG.e to Circle_demand.w
arrow from Box_PG.e to Circle_project.w
arrow from Box_PG.e to Circle_develop.n
arrow from Box_PCB.n to Circle_product.e
arrow dotted from Box_TCB.n to Box_TM.s
arrow dotted from Circle_decision.s to Box_PCB.n
arrow dotted from Circle_demand.e to Box_TCB.w
arrow dotted from Circle_develop.e to Box_PCB.s
arrow
dotted from
Circle_project.e to Box_PCB.w
arrow dotted from Circle_product.e to Box_TCB.s
arrow dotted from Circle_product.w to Box_WG.s
arrow dotted from Circle_product.w to Box_PG.n
arrow dashed from Box_PCB.w +(0,.2) to Circle_project.e +(0,.2)
arrow dashed from Box_PCB.s +(-.4,0) to Circle_develop.e +(0,.2)
arrow dashed from Box_TCB.s +(-.2,0) to Circle_product.e +(0,.2)
arrow dashed from Box_TCB.w +(0,.2) to Circle_demand.e +(0,.2)
# spline
spline -> left 5.5 from Circle_develop.w then up 8.5 \
then right 1.5
 .PE
```

Example 2

We will now draw a Christmas greeting:



The Christmas greeting was produced from the following input:

```
 .PS 3.9□
define fir Y FIR: [□
xshift=1.0i□
yshift=-2.0i□
rd=1.5i□
pi=atan2(0,-1)□
p2=pi/2□
an=pi/5□
ic=pi/60□
for i = p2 + ic to p2 + an by ic do X□
         arc -> from xshift + 0,yshift + rd \□
         to xshift + rd * cos(i),yshift + rd * sin(i) rad rd X□
# 2nd (lower) branch□
```

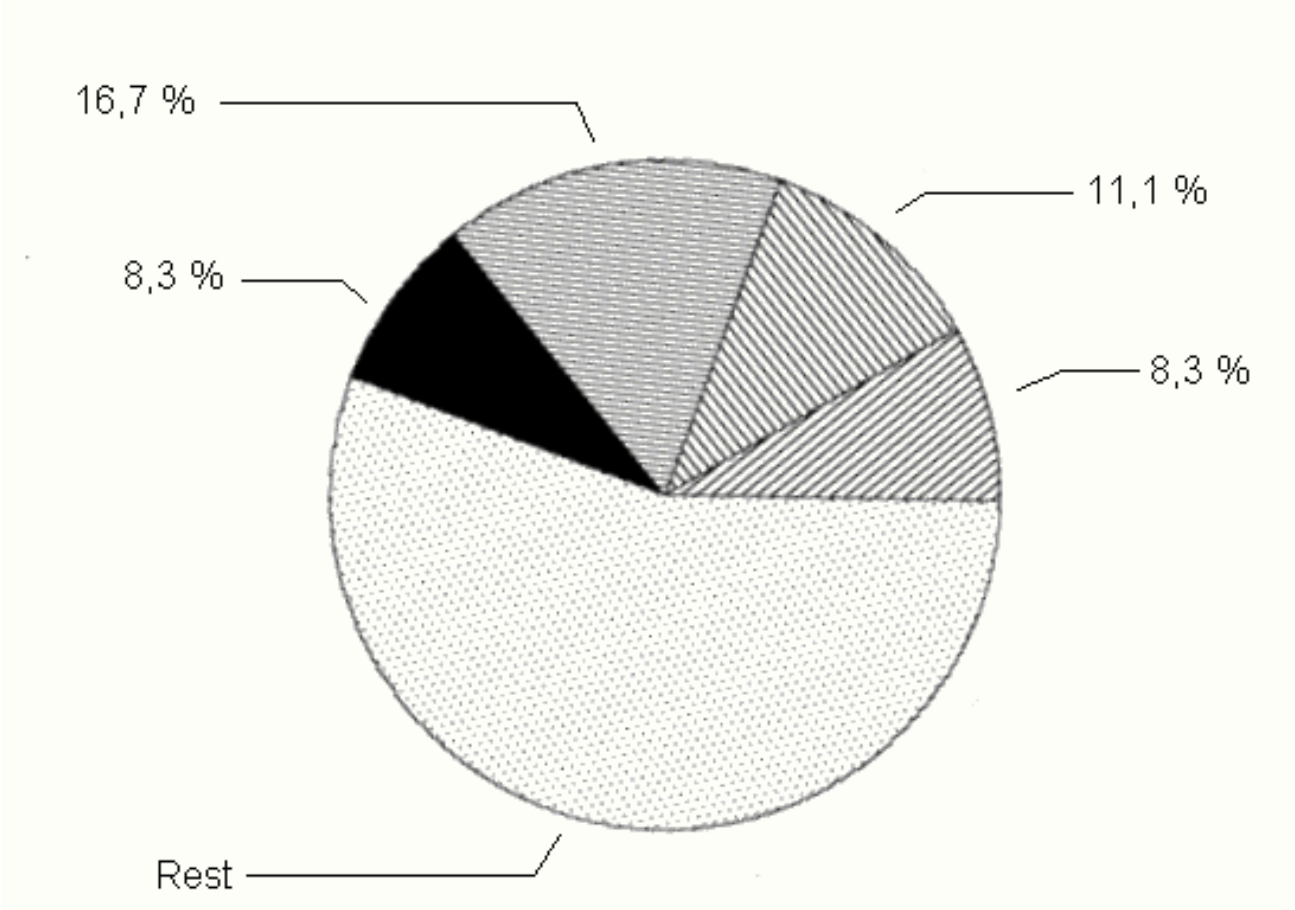
```
an2=an/2□
betan=pi/6□
r2=rd/2□
ic2=pi/30□
ctx1=rd*cos(p2+an2)□
ctx=xshift + ctx1□
cty1=rd*sin(p2+an2)□
cty=yshift+cty1-r2□
for□
i = p2
- betan + ic2 to p2 by ic2 do X□
arc -> from ctx + r2 * cos(p2 - betan),cty + r2 * sin(p2 -betan) to \□
ctx + r2 * cos(i),cty + r2 * sin(i) rad r2 X□
# glitter ball□
kh = 0.3i□
ll=kh/2□
rk= ll/2□
cx= ctx□
cy= cty+r2-ll-rk□
h=rk/5□
L: line down ll□
C: circle rad rk at cx,cy□
for ynew=0.0 to rk by h do X□
         yh=cy+ynew□
         yl=cy-ynew□
         xnew=sqrt(rk*rk-ynew*ynew)□
         xr=cx+xnew□
         xl=cx-xnew□
         line dotted .02 from xl,yh to xr-.01,yh□
  □

    line dotted .02 from xl,yl to xr-.01,yl□
X□
# 3rd (higher) branch□
an2 = 3 * an/4□
betan = pi/6□
shan = pi - pi/6□
r2 = rd/2□
ic2 = pi/30□
ctx1 = rd * cos(p2 + an2)□
ctx = xshift + ctx1 - r2 * cos(shan)□
cty1 = rd * sin(p2 + an2)□
cty = yshift + cty1 - r2 * sin(shan)□
for i = shan - betan + ic2 to shan by ic2 do X□
    arc -> from ctx + r2 * cos(shan - betan),cty + r2 * sin(shan-betan) \□
    to ctx + r2 * cos(i),cty + r2 * sin(i) rad r2 X□
# candle and wick□
kh = 0.3i□
kw = kh/4□
krzx = xshift + rd * cos(p2 + an/3)□
krzy = yshift + rd * sin(p2 + an/3)□
move□
to
krzx,krzy + kh□
```

```
KRZ: box ht kh wid kw□
line up kh/10 from KRZ.n□
# flame□
fh = kh/3□
fw = fh/3□
ellipse ht fh wid fw□
# rays□
flcx = krzx + kw/2 -.035□
flcy = krzy + kh + kh/10 + fh/2□
gamfac = 6□
gams = 2 * pi/gamfac□
shni = fh□
shno = fh * 1.8□
for i = -pi/2 + gams to 3 * pi/2 - gams by gams do X□
         line from flcx + shni * cos(i),flcy + shni * sin(i) \□
         to flcx + shno * cos(i),flcy + shno * sin(i) X□
]□
# frame□
air = 0.2i□
box□
ht
last [].ht + 1.3 * air wid last [].wid + air at last []□
Y□
fir□
 .PE
```

Example 3

Pie chart:



51

.P The pie chart was produced from the following input:

```
 .PS□
scale=2.54□
copy "pietxtmacro"□
PIE: [□
pietxt(2.5,30,0,5,0,"8.3 %")□
pietxt(2.5,40,30,15,0,"11.1 %")□
pietxt(2.5,60,70,25,0,"16.7 %")□
pietxt(2.5,30,130,490,0,"8.3 %")□
pietxt(2.5,200,160,13,1,"Rest")□
]□
circle rad 2.53 at PIE.c□
 .PE
```

The integrated `pietxtmacro` file contains the basic definitions and has the following content:

```
 .\"=========================================================□
 .\"□
 .\" pietxt: Draw a pie chart, optional dotted lines.□
 .\"         Add a string to each piece.□
 .\"□
 .\" The name of the macro is pietxt.□
 .\" Auxiliary macros are pie, pieaux, pietxaux.□
 .\"=========================================================□
 .\"□
```

```
    Parameters are:
 .\"
 .\"       cx    x-coordinate of the center of the circle.
 .\"       cy    y-coordinate of the center of the circle.
 .\"       rd    Radius of the circle.
 .\"       phi   Angle size of the piece of pie ( >=0,
 .\"             360 = full circle).
 .\"       alfa  Rotating angle of the piece (arbitrary,
 .\"             alfa=0: Start right, horizontal).
 .\"       h     Number of lines per radius to be used for
 .\"             hatching (h=0: Blank piece). Large h causes
 .\"             blackening of the piece.
 .\"       dflag =1: line is dotted. Any other value: line is thru.
 .\"

    str   A string attached to the piece of pie.
 .\"============================================================
 .\"
 .PS
#============================================================
# Constants
and
initializations
#============================================================
pi = atan2(0,-1)
pi2 = pi / 2.0
#============================================================
# pieaux(
cx,
cy, rd, phi, alfa, hdist, dfl ):
#     Auxiliary macro for  0 <= phi <= pi.
#
# phi, alfa are in radians (2 pi = full circle),
# hdist is the (absolute) distance between hatching lines.
# dfl is a flag that decides if lines are to be dotted.
#============================================================
define pieaux @
#------------------------------------------------------------
# Center
and
radius of circle
#------------------------------------------------------------
cxx = $1
cyy = $2
rdd = $3
#------------------------------------------------------------
# Angles in radians (i.e. 2 pi = full circle)
#------------------------------------------------------------
phix = $4
alfx = $5
#------------------------------------------------------------
# Distance between lines
#------------------------------------------------------------
```

```
hdist = $6
#-----------------------------------------------------------
# Line-dotting flag
#-----------------------------------------------------------
dfl = $7
#-----------------------------------------------------------
# Trigonometric
quantities
that are often used
#-----------------------------------------------------------
cosp = cos(phix)
sinp = sin(phix)
tanp = sinp / cosp
rcosp = rdd * cosp
cosa = cos(alfx)
sina = sin(alfx)
#===========================================================
# Action
#===========================================================
#-----------------------------------------------------------
# 0 <= phid <= 90
degree
#-----------------------------------------------------------
for
x = 0.0 to rdd-.01 by hdist do X
  if x < rcosp then Y yh = x * tanp Y \
                else Y yh = sqrt(rdd * rdd + 0.1 - x * x) Y
  yl = 0
#-----------------------------------------------------------
# Turn points (x,yh) and (x,yl) by alfx (2 x 2 matrix)
#-----------------------------------------------------------
  x1 = cxx + x * cosa - yh * sina
  y1 = cyy + x * sina + yh * cosa
  x2 = cxx + x * cosa - yl * sina
  y2 = cyy + x * sina + yl * cosa
  if dfl == 1 then Q line dotted hdist
from x1-.01,y1-.01 \
              to x2+.01,y2+.01
Q \
              else Q line from x1+.01,y1 to x2,y2 Q
X
#-----------------------------------------------------------
# 90 < phi <= 180 degree
#-----------------------------------------------------------
if phix > pi2 then Z
for xt = hdist to -rcosp by hdist do X
  x  = -xt
  yh = sqrt(rdd * rdd - x * x)
  yl = x * tanp
#-----------------------------------------------------------
# Turn points (x,yh) and (x,yl) by alfx (2 x 2 matrix)
#-----------------------------------------------------------
  x1 = cxx + x * cosa - yh * sina
  y1 = cyy + x * sina + yh * cosa
```

```
 □
x2 = cxx + x * cosa
- yl * sina□
  y2 = cyy + x * sina + yl * cosa□
  if dfl == 1 then Q line dotted hdist from x1-.01,y1-.01 \□
               to x2+.01,y2+.01 Q \□
               else Q line from x1,y1 to x2,y2 Q□
X□
Z□
@□
#====================== end of pieaux ====================□
#=========================================================□
# pietxaux(□
cx,
cy, rd, phi, alfa, str ): Pie text macro.□
#□
#     str ($6) is a string to be attached to the piece of pie.□
#□
#     phi and alfa are degrees (full circle = 360°).□
#=========================================================□
#---------------------------------------------------------□
# Draw a radial line for the text□
#---------------------------------------------------------□
define□
pietxaux &□
ccxx = $1□
ccyy = $2□
rrdd = $3□
phii = $4□
alff = $5□
whspace = rrdd / 8□
rd1 = rrdd
 + whspace□
rd2 = rd1 + rrdd / 10□
bigr = rd2 + ( rrdd / 20 )□
#---------------------------------------------------------□
# Compute the 'center' angle and convert it to radians□
#---------------------------------------------------------□
gamma = ( alff + phii / 2 ) * pi / 180.0□
cosg = cos( gamma )□
sing = sin( gamma )□
xx1 = ccxx + rd1 * cosg□
yy1 = ccyy + rd1 * sing□
xx2 = ccxx + rd2 * cosg□
yy2 = ccyy + rd2 * sing□
line□
from xx1,yy1 to xx2,yy2□
#---------------------------------------------------------□
# Draw a horizontal line for the text, left or right.□
# Add
text (1 string).□
#---------------------------------------------------------□
if cosg < 0 then X□
  line to ccxx - bigr,yy2□
```

```
  line invis to ccxx - bigr - whspace,yy2
  $6 rjust
X \
  else X
  line to ccxx + bigr, yy2
  line invis to ccxx + bigr + whspace,yy2
  $6 ljust
X
&
#==================== end of pietxaux ====================
#=========================================================
# pie(
cx,
cy, rd, phi, alfa, haux, dflag ):
#     Pie chart macro, raw.
#
#     Draw a circle at (cx,cy) with radius rd.
#     Cut a piece of pie of angle phi turned by alfa.
#     h is the number of lines used for hatching.
#     dflag is a line-dotting flag (=1).
#
#     phi and alfa are degrees (full circle = 360°).
#=========================================================
define
pie %
#---------------------------------------------------------
# Center
and radius of circle
#---------------------------------------------------------
cx = $1
cy = $2
rd = $3
#---------------------------------------------------------
# Angles for pie (in degrees, i.e. 360° = full circle).
# phid: Size of piece ( >= 0 ).
# alfad: Angle by which the piece is turned (arbitrary).
#---------------------------------------------------------
phid  = $4
alfad = $5
#---------------------------------------------------------
# haux: Number of lines for hatching.
# h:    Distance between lines.
#---------------------------------------------------------
haux = $6
h = 2 * rd
if
haux > 0
then W
  h = rd / haux
W
#---------------------------------------------------------
# Line-dotting flag
#---------------------------------------------------------
dflag = $7
```

```
#-------------------------------------------------------□
# Angles in radians (i.e. 2 pi = full circle)□
#-------------------------------------------------------□
phi = phid * pi / 180.0□
alfa = alfad * pi / 180.0□
beta = alfa + phi□
#=======================================================□
# Action□
#=======================================================□
#-------------------------------------------------------□
# Draw□
circle.□
# Delimit
the piece of pie by lines.□
#-------------------------------------------------------□
circle rad rd at cx,cy□
circle rad rd+.01 at cx,cy□
line from cx,cy to cx + rd * cos(alfa),cy + rd * sin(alfa)□
line from cx,cy to cx + rd * cos(beta),cy + rd * sin(beta)□
#-------------------------------------------------------□
# Hatch only if haux > 0 (number of lines for hatching).□
#-------------------------------------------------------□
if haux > 0 then Y□
#-------------------------------------------------------□
# 0 <= phid <= 180 degree□
#-------------------------------------------------------□
if phi <= pi then W□
 □
pieaux( cx, cy, rd, phi, alfa,
h, dflag )□
W□
#-------------------------------------------------------□
# phid > 180 degree□
#-------------------------------------------------------□
if phi > pi then W□
  pieaux( cx, cy, rd, pi, alfa, h, dflag )□
  pieaux( cx, cy, rd, phi - pi, alfa + pi, h, dflag )□
W□
Y□
%□
#==================== end of pie ====================□
#=======================================================□
# pietxt(□
rd,
phi, alfa, h, dflag, str ):□
#     Pie chart macro, with text.□
#□
#     Draw a circle with radius rd.□
#     Cut a piece of pie of angle phi turned by alfa.□
#     h is the number of lines used for hatching.□
#     dflag is a line-dotting flag (=1).□
#     str is a string to be attached to the piece of pie.□
#□
#     phi and alfa are degrees (full circle = 360°).□
```

```
#=======================================================
define
pietxt
!
pie( 0, 0, $1, $2, $3, $4, $5 )
pietxaux( 0, 0, $1, $2, $3, $6 )
!
#====================== end of pietxt ====================
 .PE
```

SEE ALSO

grap(1M), troff(1M), troff(5).

REFERENCES

B. W. Kernighan, *PIC – A Crude Graphics Language for Typesetting*

D. Dougherty, T. O'Reilly, *UNIX TEXT PROCESSING*, Hayden Books, Indianapolis, Indiana 46268 USA, ISBN 0-672-46291-5

System Administrator's Reference Manual

picpack(1M)
Line Printer PostScript Utilities

NAME

`picpack -` PostScript picture packing preprocessor

SYNOPSIS

`/usr/lib/lp/postscript/picpack` [*options*] [*files*]

DESCRIPTION

`picpack` copies *files* to stdout, expanding picture inclusion requests (marked by the `.BP` or `.PI` macros) into an inline format that can be passed through `troff` and handled by `dpost`. If no *files* are specified or if – is one of the input *files*, standard input is read.

OPTIONS

–k*list*

list is a comma or space separated string of words used to locate picture inclusion requests. The start of every line in the input *files* is compared with each word in *list*. If there's a match, the second string on the line is taken as the pathname of a picture file that's added to the output file. The default *list* is ".`BP` `.PI`".

`–q` Suppress "missing picture file" error messages.

`picpack` is a trivial preprocessor that, in a sense, duplicates some of the picture inclusion capabilities already available in `dpost`. `picpack` should not be used if your formatting command line includes a call to `dpost`. Its only purpose is to combine picture files with text in a single file that can be passed through `troff` and unpacked, at some later time, by `dpost`. The original picture inclusion mechanism, with files are pulled in by `dpost`, is the preferred approach.

EXAMPLES

A typical application might be in a distributed printing environment where everything up to `troff` is run by the user and everything after `troff` is handled by a spooling daemon (perhaps `lp`). In that case the command line would be,

```
pic file | tbl | eqn | picpack | troff -mm -Tpost | lp
```

A bad example, although one that should still work, would be,

```
pic file | tbl | eqn | picpack | troff -mm -Tpost | dpost >file.ps
```

In this case picture inclusion requests could (and should) be handled by `dpost`. Running `picpack` is not needed or even recommended. It should be dropped from any pipeline that includes a call to `dpost`.

WARNINGS

Combining pictures and text using the capabilities available in `dpost` is the recommended approach and is always guaranteed to be more efficient than `picpack`. Running `picpack` and `dpost` in the same pipeline makes little sense.

Using `picpack` will likely result in files that can no longer be reliably passed through other important `troff` postprocessors like `proof`. At present `picpack` is only guaranteed to work with `dpost`.

DIAGNOSTICS

0 exit status is returned if *files* were successfully processed.

SEE ALSO

`dpost`(1M) , `troff`(1M) .

System Administrator's Reference Manual

pkgadd(1M)

NAME

`pkgadd` - transfer software package to the system

SYNOPSIS

`/usr/sbin/pkgadd` [`–d` *device*] [`–r` *response*] [`–l`|`–q`] [`–n`|`–O`|`–c`[`h`]] [`–a` *admin*]
□□□[*pkginst1* [*pkginst2*[...]]]

`/usr/sbin/pkgadd –s` *spool* [`–d` *device*] [*pkginst1* [*pkginst2*[...]]]

DESCRIPTION

`pkgadd` transfers the contents of a software package from the distribution medium or directory to install it onto the system. Used without the `–d` option, `pkgadd` looks in the default spool directory for the package (`/var/spool/pkg`). Used with the `–s` option, it reads the package to a spool directory instead of installing it. If *pkginst* is not specified, all packages available under *device* are offered for selection in a menu.

OPTIONS

`–d` Installs or copies a package from *device*. *device* can be a full path name to a directory or the identifier for tape (for example `/var/tmp`). It can also be the device alias.

`–r` Identifies a file or directory, *response*, which contains output from a previous `pkgask` session. This file supplies the interaction responses that would be requested by the package in interactive mode. *response* must be a full pathname.

`–l` Log error messages in `/var/sadm/install/logs`.

`–q` No messages (quiet mode).

`–n` Non-interactive mode. The command is aborted should interaction be required. Interactive input should only occur in `request` scripts. A `request` script without a *response* will likewise cause an abort. `pkgadd` cannot recognize interactive input in installation scripts, though the situation should not arise.

`–O` The menu for *pkginst*'s is only displayed once.

`–c` Check mode (virtual install). The packages are not actually added, rather a check is run to see whether an installation could be carried out successfully (dependencies, disk space etc.).

`–ch` Records the information from a virtual install for other virtual installations.

`–a` Defines an installation administration file, *admin*, to be used in place of the default administration file. The token `none` overrides the use of any *admin* file, and thus forces interaction with the user. Unless a full path name is given, `pkgadd` looks in the `/var/sadm/install/admin` directory for the file.

`–s` Reads the package into the directory *spool* instead of installing it.

pkginst
Specifies the package instance or list of instances to be installed. The token `all` may be used to refer to all packages available on the source medium. The *pkginst*.* format can be used to indicate all instances of a package.

When executed without options, `pkgadd` uses `/var/spool/pkg` (the default spool directory).

DIAGNOSTICS

The following return codes exist:

Code	Meaning
0	Successful execution
1	Abort (usually a wrong call)

2	Warning (e.g. only partly installed)
3	User interrupt
4	Abort because of `admin` entry
5	Abort; interaction required
+10	Reboot required when all packages are installed
+20	Reboot required immediately
NOSET (77)	No package selected (for Set Installation Packages only)
>=79	Aborted by pkg-internal error:
	Device errors:
79	Device not available
80	Device not in device table
	Data stream errors:
81	No archive in data stream
82	Cannot initialize data stream device
83	Error in `ds_next()`
84	Data stream expected
	Package errors:
85	Package list empty
86	`pkgmap` or `pkginfo` missing
87	Response file cannot be created/deleted
88	Error opening `pkgmap`|`pkginfo`|`depend`|`admin`
89	Error opening the response file
90	Package sections <1
91	Faulty processing of `pkgmap`
92	Corrupted file in package
93	Directory structure corrupted
94	`EXISTING_VERSION` of the package has not been installed
95	`EXISTING_LOAD` of the package has not been installed
96	`pkgmap` entries and `contents` entries are not consistent
97	`CLASSES` parameter not defined
	System errors:
99	No more memory
100	`ulimit` error
101	Working directory cannot be identified
102	Pipe error
103	System run state cannot be determined
104	`pkginfo` error

105	`uncompress` error
106	Lock error
107	`umount` impossible
108	`fseek()` error
	`contents` file errors:
110	Error opening the `contents` file
111	Incorrect entry in the `contents` file
112	Incorrect status of an entry in the `contents` file
113	Error writing to the `contents` file
	Directory errors:
116	Cannot create directory
117	Cannot switch to directory
118	Cannot copy to directory
119	Cannot delete directory
120	Cannot create temporary file
	Pyramid functions errors:
121	Error in `__pyr_save_cp`
122	Error in `__pyr_copy_files`
123	Error in `__pyr_arm`

If more than one package is installed, the return code applies to the package installed last.

NOTES

When transferring a package to a spool directory, the `-r`, `-n`, and `-a` options cannot be used.

The `-r` option can be used to indicate a directory name as well as a filename. The directory can contain numerous *response* files, each sharing the name of the package with which it should be associated. This would be used, for example, when adding multiple interactive packages with one invocation of `pkgadd`. Each package would need a *response* file. If you create response files with the same name as the package (i.e. *package1* and *package2*), then name the directory in which these files reside after the `-r`.

The `-n` option will cause the installation to halt if any interaction is needed to complete it.

Packages compressed using `pkgmk -c` are decompressed automatically.

SEE ALSO

`pkginfo`(1M), `pkgmk`(1M), `pkgrm`(1M).

System Administrator's Reference Manual

pkgask(1M)

NAME

`pkgask` - store answers to a request script

SYNOPSIS

`/usr/sbin/pkgask` [`-d` *device*] `-r` *response pkginst1* [*pkginst2* [...]]

DESCRIPTION

`pkgask` allows the administrator to store answers to an interactive package (one with a request script). Invoking this command generates a *response* file that is then used as input at installation time. The use of this *response* file prevents any interaction from occurring during installation since the file already contains all of the information the package needs.

OPTIONS

`-d` Runs the request script for a package on *device*. *device* can be a full path name to a directory or the identifier for tape (for example `/var/tmp`). It can also be the device alias. The default device is the installation spool directory.

`-r` Identifies a file or directory, which should be created to contain the responses to interaction with the package. The name must be a full pathname. The file, or directory of files, can later be used as input to the `pkgadd` command.

pkginst
Specifies the package instance or list of instances for which request scripts will be created. The token `all` may be used to refer to all packages available on the source medium.

NOTES

The `-r` option can be used to indicate a directory name as well as a filename. The directory name is used to create numerous *response* files, each sharing the name of the package with which it should be associated. This would be used, for example, when you will be adding multiple interactive packages with one invocation of `pkgadd`. Each package would need a *response* file. To create multiple response files with the same name as the package instance, name the directory in which the files should be created and supply multiple instance names with the `pkgask` command. When installing the packages, you will be able to identify this directory to the `pkgadd` command.

SEE ALSO

`installf`(1M), `pkgadd`(1M), `pkgchk`(1M), `pkgmk`(1M), `pkginfo`(1M), `pkgparam`(1M), `pkgproto`(1M), `pkgtrans`(1M), `pkgrm`(1M), `removef`(1M).

System Administrator's Reference Manual

pkgchk(1M)

NAME

pkgchk - check accuracy of packages

SYNOPSIS

/usr/sbin/pkgchk [−l|−L|−acfqv] [−n|−N] [−x] [−p *path1*[,*path2 ...*]] [−i *file*] [*pkginst ...*]

/usr/sbin/pkgchk −d *device* [−l|−L|−v] [−p *path1*[,*path2 ...*]] [−i *file*] [*pkginst ...*]

/usr/sbin/pkgchk -m *pkgmap* [−e *envfile*] [−l|−L|−acfqv] [−n|−N] [−x] [−i *file*] [−p *path1*[,*path2 ...*]]

DESCRIPTION

pkgchk checks the accuracy of package objects. The command checks the integrity of directory structures and the files. Discrepancies are reported on stderr along with a detailed explanation of the problem. If either the −l option or the −L option is used, a list containing information on package objects is output in a list. A check is not run.

The first synopsis defined above is used to check or list the contents and/or attributes of objects that are currently installed on the system. Package names may be listed on the command line, or by default the entire contents of a machine will be checked.

The second synopsis is used to check or list the contents of a package which has been spooled on the specified device, but not installed. Note that attributes cannot be checked for spooled packages.

The third synopsis is used to check or list the contents and/or attributes of objects which are described in the indicated *pkgmap*.

OPTIONS

−a Audits the file attributes only, does not check file contents. Default is to check both.

−c Audits the file contents only, does not check file attributes. Default is to check both.

−f Corrects file attributes if possible. If used with the −x option, it removes hidden files. When pkgchk is invoked with this option it creates directories, named pipes, links and special devices if they do not already exist.

−q Quiet mode. Does not give messages about missing files.

−v Verbose mode. Files are listed as processed.

−l Lists information on the selected files that make up a package. A check is not run.

It is not compatible with the −a, −c, −f, −g, −v, and −L options.

−L Lists information on the selected files that make up a package as they are entered in the contents or pkgmap file. A check is not run.

It is not compatible with the −a, −c, −f, −g, −v, and −l options.

−n Volatile or editable files are not checked. This should be used for most post-installation checking.

−N Volatile or editable files are always checked.

−x Searches exclusive directories, looking for files which exist that are not in the installation software database or the indicated *pkgmap* file.

−p Only checks the accuracy of the pathname or pathnames listed. *pathname* can be one or more pathnames separated by commas (or by white space, if the list is quoted).

−i Reads a list of pathnames from *file* and compares this list against the installation software database or the indicated *pkgmap* file. Pathnames which are not contained in *inputfile* are not checked.

−d Specifies the device on which a spooled package resides. *device* can be a full path name to a directory

or the identifier for tape (for example `/var/tmp`). It can also be the device alias.

`-m` Requests that the package be checked against the pkgmap file *pkgmap*.

`-e` Requests that the pkginfo file named as *envfile* be used to resolve parameters noted in the specified pkgmap file.

pkginst

Specifies the package instance or instances to be checked. The format *pkginst*`.*` can be used to check all instances of a package. The default is to display all information about all installed packages.

SEE ALSO

`pkgadd`(1M), `pkgask`(1M), `pkginfo`(1M), `pkgrm`(1M), `pkgtrans`(1M).

System Administrator's Reference Manual

pkgeksrm(1M)

NAME

pkgeksrm - remove single correctional release (SCR) packages fromsystem

SYNOPSIS

/usr/sbin/pkgeksrm [–n | –b *basepkg*] *scr_inst* [...]

DESCRIPTION

pkgeksrm removes previously installed SCR packages from the system.

Several SCR *scr_inst* packages can be specified. Wildcards can also be used within *scr_inst*, which must be canceled with single quotes ('...') or with a backslash (\).

The files belonging to the packages are not deleted because the command should only be used directly before an update installation.

OPTIONS

–n (no remove) The SCR packages are not removed; you are only informed of the SCR packages that would have been removed. If the –n option is specified, any –b option is ignored.

–b *basepkg*

When removing an SCR package, pkgeksrm uses this option to check whether a package information file eksinfo exists for this SCR, which contains a

BASEPKG=*basepkg*

parameter. If the file does not exist, or if this parameter is missing, or if the parameter value does not match the *basepkg* options argument, the removal of the SCR package will be rejected.

If this option is not specified, none of these checks will be carried out.

DIAGNOSTICS

The following return codes should be noted:

0 Successful execution

1 Abort (usually a wrong call)

2 Warning

3 User interrupt

85 No package *scr_inst* ... available

88 Error opening eksinfo file

94 BASEPKG entry in eksinfo file is incorrect

NOTES

This command should be used with care; since it is not possible to determine whether a package is an SCR package, you could also remove normal packages from the system with pkgeksrm.

FILES

/var/sadm/pkg/*pkginst*/install/eksinfo

SEE ALSO

pkginfo(1M), pkgrm(1M), removef(1M).

System Administrator's Reference Manual

pkginfo(1M)

NAME

`pkginfo` - display software package information

SYNOPSIS

`pkginfo` [`-q|x|l|L`] [`-p|i`] [`-N`] [`-k`] [`-a`*arch*] [`-v` *version*] [`-c` *category1*[,*category2*[, ...]]] [*pkginst*[,*pkginst*[, ...]]]

`pkginfo -d` *device* [`-q|x|l|L`] [`-N`] [`-a` *arch*] [`-v` *version*] [`-c` *category1*[,*category2*[, ...]]] [*pkginst*[,*pkginst*[, ...]]]

DESCRIPTION

`pkginfo` displays information about software packages which are installed on the system (with the first synopsis) or which reside on a particular device or directory (with the second synopsis). Only the package name and abbreviation for pre-SVR4 packages will be included in the display.

OPTIONS

`-q` Does not list any information, but can be used from a program to check (i.e. query) whether or not a package has been installed.

This option is only worthwhile if a single package instance is specified.

`-x` Designates an extracted listing of package information. It contains the package abbreviation, package name, package architecture (if available) and package version (if available).

`-l` Designates long format, which includes all available information about the designated package(s).

`-L` Name of *pkginst* only.

`-p` Designates that information should be presented only for partially installed packages.

`-i` Designates that information should be presented only for fully installed packages.

`-N` No pre-SVR4 packages.

`-k` Check mode. Virtually installed and deinstalled packages (see `pkgadd`, `pkgrm -ch`) are taken into account. The `-k` option is not compatible with the `-d`, `-l` and `-c` options.

`-a` Specifies the architecture of the package as *arch*.

`-v` Specifies the version of the package as *version*. "All compatible versions" can be requested by preceding the version name with a tilde (`~`). Multiple white space is replaced with a single space during version comparison.

`-c` Selects packages to be display based on the category *category*. (Categories are defined in the category field of the `pkginfo` file.) If more than one category is supplied, the package must only match one of the list of categories. The match is not case specific.

`-d` Defines a device, *device*, on which the software resides. *device* can be a directory pathname or the identifiers for tape, floppy disk, removable disk, etc. The special token "spool" may be used to indicate the default installation spool directory.

pkginst

Designates a package by its instance. An instance can be the package abbreviation or a specific instance (for example, `inst.1` or `inst.beta`). All instances of package can be requested by `inst.*`.

NOTES

Without options, `pkginfo` lists the primary category, package instance, and name of all completely installed and partially installed packages. One line per package selected is produced.

The `-p` and `-i` options are meaningless if used in conjunction with the `-d` option.

The options –q, –x, –l, and –L are mutually exclusive.

The –k option is not compatible with the –d, –l and –c options.

pkginfo cannot tell if a pre-SVR4 package is only partially installed. It is assumed that all packages are fully installed.

SEE ALSO

pkgadd(1M), pkgask(1M), pkgchk(1M), pkgrm(1M), pkgtrans(1M).

System Administrator's Reference Manual

pkgmk(1M)

NAME

pkgmk - produce an installable package

SYNOPSIS

pkgmk [−oc0] [−d *device*] [−r*rootpath*] [−b *basedir*] [−l *limit*] [−a *arch*] [−v *version*] [−p *pstamp*] [−f *prototype*] [*variable=value ...*] [*pkginst*]

DESCRIPTION

pkgmk produces an installable package. The package is stored in file system format [see pkgtrans(1M)].

The command uses the package prototype file as input and creates a pkgmap file. The contents for each entry in the prototype file are copied to the appropriate output location. Information concerning the contents (checksum, file size, modification date) is computed and stored in the pkgmap file, along with attribute information specified in the prototype file.

Pathnames in a call to pkgmk must be in absolute form.

OPTIONS

−o Overwrites the same instance, package instance will be overwritten if it already exists.

−c The package will be compressed by the compress(1) command. Only information files are not compressed.

−O Optimizes the compression via −c: the files are compressed while they are copied to the output medium. Otherwise compressing is a separate step. −O is not possible for split packages.

−d Creates the package on *device*. *device* can be a full path name to a directory or the identifier for tape (for example /var/tmp). It can also be the device alias. The default device is the installation spool directory.

−r Ignores destination paths in the prototype file. Instead, uses the indicated *rootpath* with the source path name appended to locate objects on the source machine.

−b Prepends the indicated *basedir* to locate relocatable objects on the source machine.

−l Specifies the maximum size in 512 byte blocks of the output device as *limit*. By default, if the output file is a directory or a mountable device, pkgmk will employ the df command to dynamically calculate the amount of available space on the output device. Useful in conjunction with pkgtrans to create package with datastream format.

−a Overrides the value of ARCH in the pkginfo file with *arch*.

−v Overrides the value of VERSION in the pkginfo file with *version*.

−p Overrides the value of PSTAMP in the pkginfo file with *pstamp*.

−f Uses the file *prototype* as input to the command. The default *prototype* file name is [Pp]rototype.

variable=value
Places the indicated variable in the packaging environment. See prototype(4) for definitions of packaging variables.

pkginst
Specifies the package by its instance. An instance can be the package abbreviation or a specific instance (for example, inst.1).

NOTES

Architecture information is provided on the command line with the −a option or in the pkginfo file. If no architecture information is supplied at all, the output of uname −m will be used.

Version information is provided on the command line with the `-v` option or in the `pkginfo` file. If no version information is supplied, a default based on the current date will be provided.

Command line definitions for both architecture and version override the `pkginfo` definitions.

SEE ALSO

`pkgparam`(1M), `pkgproto`(1M), `pkgtrans`(1M).

System Administrator's Reference Manual

pkgparam(1M)

NAME

`pkgparam` - displays package parameter values

SYNOPSIS

`pkgparam` [–v] [–d *device*] *pkginst* [*param* [...]]

`pkgparam` –f *file* [–v] [*param* [...]]

DESCRIPTION

`pkgparam` displays the value associated with the parameter or parameters requested on the command line. The values are located in either the `pkginfo` file for *pkginst* or from the specific file named with the –f option.

One parameter value is shown per line. Only the value of a parameter is given unless the –v option is used. With this option, the output of the command is in this format:

parameter1='*value1*'
parameter2='*value2*'
parameter3='*value3*'

If no parameters are specified on the command line, values for all parameters associated with the package are shown.

OPTIONS

–v Specifies verbose mode. Displays name of parameter and its value.

–d Specifies the *device* on which a *pkginst* is stored. *device* can be a full path name to a directory or the identifier for tape (for example `/var/tmp`). It can also be the device alias. The default device is the installation spool directory. If no instance name is given, parameter information for all packages residing in *device* is shown.

–f Requests that the command read *file* for parameter values.

pkginst
Defines a specific package instance for which parameter values should be displayed. The format *pkginst*.* can be used to indicate all instances of a package.

param
Defines a specific parameter whose value should be displayed.

DIAGNOSTICS

If parameter information is not available for the indicated package, the command exits with a non-zero status.

NOTES

The –f synopsis allows you to specify the file from which parameter values should be extracted. This file should be in the same format as a `pkginfo` file. As an example, such a file might be created during package development and used while testing software during this stage.

SEE ALSO

`pkgmk`(1M) , `pkgproto`(1M) , `pkgtrans`(1M) .

System Administrator's Reference Manual

pkgproto(1M)

NAME

pkgproto - generate a prototype file

SYNOPSIS

pkgproto [−inx] [−c *class*] [*path1*[=*path2*] ...]

DESCRIPTION

pkgproto scans the indicated paths and generates a prototype on stdout that may be used as input to the pkgmk command.

Pathnames in a call to pkgproto must be in absolute form.

OPTIONS

−i Ignores symbolic links and records the paths as *ftype*=f (a file) versus *ftype*=s (symbolic link).

−c Maps the class of all paths to *class*.

−n Only *path2* is entered in the prototype when calling with *path1*=*path2* (except for links).

−x Attaches size, checksum, and modification time to every entry, i.e. output similar to pkgmap is created.

path1

Pathname where objects are located.

path2

Pathname which should be substituted on output for *path1*.

If no paths are specified on the command line, standard input is assumed to be a list of paths. If the pathname listed on the command line is a directory, the contents of the directory is searched. However, if input is read from stdin, a directory specified as a pathname will not be searched.

EXAMPLE

The following two examples show uses of pkgproto and a partial listing of the output produced.

Example 1:

```
$ pkgproto /bin=bin /usr/bin=usrbin /etc=etc
f none bin/sed=/bin/sed 0775 bin bin
f none bin/sh=/bin/sh 0755 bin daemon
f none bin/sort=/bin/sort 0755 bin bin
f none usrbin/sdb=/usr/bin/sdb 0775 bin bin
f none usrbin/shl=/usr/bin/shl 4755 bin bin
d none etc/master.d 0755 root daemon
f none etc/master.d/kernel=/etc/master.d/kernel 0644 root daemon
f none etc/rc=/etc/rc 0744 root daemon
```

Example 2:

```
$ find / −type d −print | pkgproto
d none / 755 root root
d none /usr 755 root root
d none /usr/bin 775 bin bin
d none /etc 755 root root
d none /tmp 777 root root
```

NOTES

By default, pkgproto creates symbolic link entries for any symbolic link encountered (*ftype*=s). When you use the −i option, pkgproto creates a file entry for symbolic links (*ftype*=f). The prototype file would have to

be edited to assign such file types as "v" (volatile), "e" (editable), or "x" (exclusive directory). A file or directory with the access rights 000 has a ? entered as the mode. If you would like to actually install the file with the mode 000, ? must be changed to 000. `pkgproto` detects linked files. If multiple files are linked together, the first path encountered is considered the source of the link.

SEE ALSO

`pkgmk`(1M) , `pkgparam`(1M) , `pkgtrans`(1M) .

System Administrator's Reference Manual

pkgrm(1M)

NAME

`pkgrm` - remove packages from the system

SYNOPSIS

`/usr/sbin/pkgrm` [`–n`|`–c`[`h`]] [`–a` *admin*] [*pkginst1* [*pkginst2*[...]]]

`/usr/sbin/pkgrm –s` *dir* [*pkginst1* [*pkginst2*[...]]]

DESCRIPTION

`pkgrm` will remove previously installed or partially installed packages from the system. A check is made to determine if any other packages depend on the one being removed. The action taken if a dependency exists is defined in the *admin* file or in the default *admin* file.

The default state for the command is in interactive mode, meaning that prompt messages are given during processing to allow the administrator to confirm the actions being taken. Non-interactive mode can be requested with the `–n` option.

The `–s` *dir* option can be used to remove spooled packages.

OPTIONS

`–n` Non-interactive mode. The command is aborted should interaction be required. `pkrm` cannot recognize interactive input in installation scripts, though the situation should not arise.

Use of this option requires that at least one package instance be named when the command is called.

`–c` Check mode (virtual deinstall). The packages are not actually removed, rather a check is run to see whether a deinstall could be carried out successfully (dependencies etc.).

`–ch` Records the information from a virtual deinstall for other virtual deinstallations.

`–a` Defines an installation administration file, *admin*, to be used in place of the default *admin* file.

`–s` Removes the specified package(s) from the *dir* directory. The `spool` alias name for the standard spool directory can also be specified for *dir*.

pkginst

Specifies the package to be removed. The *pkginst*.* format can be used to remove all instances of a package.

NOTES

When you are removing installed packages, you will need approx. 3 MB capacity in the `/var` directory. Please ensure that this space is available before invoking the command.

SEE ALSO

`installf`(1M), `pkgadd`(1M), `pkgask`(1M), `pkgchk`(1M), `pkginfo`(1M), `pkgmk`(1M), `pkgparam`(1M), `pkgproto`(1M), `pkgtrans`(1M), `removef`(1M).

System Administrator's Reference Manual

pkgtrans(1M)

NAME

pkgtrans - translate package format

SYNOPSIS

pkgtrans [–ionsc] *device1 device2* [*pkginst1* [*pkginst2* [...]]]

DESCRIPTION

pkgtrans translates installable packages from one format to another. It translates:

- afile system format to a datastream
- adatastream to a file system format
- afile system format to another file system format

If the target is a datastream, one datastream is generated from all translated package instances.

OPTIONS

–i Copies only the pkginfo and pkgmap files.

–o Overwrites the same instance on the destination device.

–n Creates a new instance if any instance of this package already exists.

–s Indicates that the package should be written to *device2* as a datastream rather than as a file system. The default behavior is to write a file system format on devices that support both formats.

–c When creating a data stream, the –H odc option is used for an internal cpio call [see cpio(1)]. This ensures exchange with external systems.

device1
Indicates the source device. The package or packages on this device will be translated and placed on *device2*.

device2
Indicates the destination device. Translated packages will be placed on this device. If *device2* is a filename for a plain file, the package is written as a datastream.

pkginst
Specifies which package instance or instances on *device1* should be translated. The token all may be used to indicate all packages. *pkginst*.* can be used to indicate all instances of a package. If no packages are defined, a prompt shows all packages on the device and asks which to translate.

NOTES

Device specifications can be either a special node name or a device alias. The device spool indicates the default spool directory. Source and destination devices may not be the same.

By default, pkgtrans will not transfer any instance of a package if any instance of that package already exists on the destination device. Use of the –n option will create a new instance if an instance of this package already exists. Use of the –o option will overwrite the same instance if it already exists. Neither of these options is useful if the destination device is a datastream.

EXAMPLES

The following example translates packages pkg1 and pkg2 on /tmp and places their translations (i.e. a datastream) on the 9track1 output device.

```
pkgtrans /tmp 9track1 pkg1 pkg2
```

The next example translates pkg1 and pkg2 on /tmp and places them on the diskette in a datastream format.

```
pkgtrans -s /tmp diskette0 pkg1 pkg2
```

SEE ALSO

cpio(1), installf(1M), pkgadd(1M), pkgask(1M), pkginfo(1M), pkgmk(1M), pkgparam(1M), pkgproto(1M), pkgrm(1M), removef(1M).

System Administrator's Reference Manual

pmadm(1M)

NAME

pmadm - port monitor administration

SYNOPSIS

pmadm −a [−p *pmtag* | −t *type*] −s *svctag* −i *id* −m *pmspecific* −v *ver* [−f xu] [−y *comment*] [−z *script*]

pmadm −r −p *pmtag* −s *svctag*

pmadm −e −p *pmtag* −s *svctag*

pmadm −d −p *pmtag* −s *svctag*

pmadm −l [−t *type* | −p *pmtag*] [−s *svctag*]

pmadm −L [−t *type* | −p *pmtag*] [−s *svctag*]

pmadm −g −p *pmtag* −s *svctag* [−z *script*]

pmadm −g −s *svctag* −t *type* −z *script*

DESCRIPTION

pmadm is the administrative command for the lower level of the Service Access Facility (SAF) hierarchy, that is, for service administration. A port may have only one service associated with it although the same service may be available through more than one port. In order to uniquely identify an instance of a service the pmadm command must identify both the port monitor or port monitors through which the service is available (−p or −t) and the service (−s). See the option descriptions below.

pmadm performs the following functions:

- add or remove a service
- enable or disable a service
- install or replace a per-service configuration script
- print requested service information

Any user on the system may invoke pmadm to request service status (−l or −L) or to print per-service configuration scripts (−g without the −z option). pmadm with other options may be executed only by a privileged user.

OPTIONS

−a Adds a service. pmadm adds an entry for the new service to the port monitor's administrative file. Because of the complexity of the options and arguments that follow the −a option, it may be convenient to use a command script or the menu system to add services. If you use the menu system, enter sysadm ports, then choose the port_services option.

−d Disables a service. Add x to the flag field in the entry for the service *svctag* in the port monitor's administrative file. This is the entry used by port monitor *pmtag*. See the −f option, below, for a description of the flags available.

−e Enables a service. Remove x from the flag field in the entry for the service *svctag* in the port monitor administrative file. This is the entry used by port monitor *pmtag*. See the −f option, below, for a description of the flags available.

−f xu

The −f option specifies one or both of the following two flags which are then included in the flag field of the entry for the new service in the port monitor's administrative file. If the −f option is not included, no flags are set and the default conditions prevail. By default, a new service is enabled and no utmp entry is created for it. A −f option without a following argument is illegal.

x Do not enable the service *svctag* available through port monitor *pmtag*.

u Creates a `utmp` entry for service *svctag* available through port monitor *pmtag*.

`-g` Prints, installs, or replaces a per-service configuration script. The `-g` option with a `-p` option and a `-s` option prints the per-service configuration script for service *svctag* available through port monitor *pmtag*. The `-g` option with a `-p` option, a `-s` option, and a `-z` option installs the per-service configuration script contained in the file *script* as the per-service configuration script for service *svctag* available through port monitor *pmtag*. The `-g` option with a `-s` option, a `-t` option, and a `-z` option installs the file *script* as the per-service configuration script for service *svctag* available through any port monitor of type *type*. Other combinations of options with `-g` are invalid.

`-i` *id*
id is the identity that is to be assigned to service *svctag* when it is started. *id* must be an entry in `/etc/passwd`.

`-l` The `-l` option requests service information. Used by itself and with the options described below it provides a filter for extracting information in several different groupings.

`-l` By itself, the `-l` option lists all services on the system.

`-l -p` *pmtag*
Lists all services available through port monitor *pmtag*.

`-l -s` *svctag*
Lists all services with tag *svctag*.

`-l -p` *pmtag* `-s` *svctag*
Lists service *svctag*.

`-l -t` *type*
Lists all services available through port monitors of type *type*.

`-l -t` *type* `-s` *svctag*
Lists all services with tag *svctag* available through a port monitor of type *type*.

Other combinations of options with `-l` are invalid.

`-L` The `-L` option is identical to the `-l` option except that output is printed in a condensed format.

`-m` *pmspecific*
pmspecific is the port monitor-specific portion of the port monitor administrative file entry for the service.

`-p` *pmtag*
Specifies the tag associated with the port monitor through which a service (specified as `-s` *svctag*) is available.

`-r` Removes a service. When `pmadm` removes a service, the entry for the service is removed from the port monitor's administrative file.

`-s` *svctag*
Specifies the service tag associated with a given service. The service tag is assigned by the system administrator and is part of the entry for the service in the port monitor's administrative file.

`-t` *type*
Specifies the the port monitor type.

`-v` *ver*
Specifies the version number of the port monitor administrative file. The version number may be given as

`-v` `` `pmspec -V` ``

where `pmspec` is the special administrative command for port monitor *pmtag*. This special command is `ttyadm` for `ttymon` and `nlsadmin` for `listen`. The version stamp of the port monitor is known by the command and is returned when `pmspec` is invoked with a `-V` option.

`-y` *comment*
Associates *comment* with the service entry in the port monitor administrative file.

−z *script*
Used with the −g option to specify the name of the file that contains the per-service configuration script. Modifying a configuration script is a three-step procedure. First a copy of the existing script is made (−g alone). Then the copy is edited. Finally, the copy is put in place over the existing script (−g with −z).

DIAGNOSTICS

If successful, pmadm will exit with a status of 0. If it fails for any reason, it will exit with a nonzero status.

NOTES

Options that request information write the requested information to the standard output. A request for information using the −l option prints column headers and aligns the information under the appropriate headings. In this format, a missing field is indicated by a hyphen. A request for information in the condensed format using the −L option prints the information in colon-separated fields; missing fields are indicated by two successive colons. # is the comment character.

EXAMPLES

Add a service to a port monitor with tag pmtag. Give the service the tag svctag. Port monitor-specific information is generated by specpm. The service defined by svctag will be invoked with identity root.

```
pmadm −a −p pmtag −s svctag −i root −m `specpm −a arg1 \
−b arg2` −v `specpm -V`
```

Add a service with service tag svctag, identity guest, and port monitor-specific information generated by specpm to all port monitors of type type:

```
pmadm −a −s svctag −i guest −t type −m `specpm −a arg1 \
−b arg2` −v `specpm -V`
```

Remove the service svctag from port monitor pmtag:

```
pmadm −r −p pmtag −s svctag
```

Enable the service svctag available through port monitor pmtag:

```
pmadm −e −p pmtag −s svctag
```

Disable the service svctag available through port monitor pmtag:

```
pmadm −d −p pmtag −s svctag
```

List status information for all services:

```
pmadm −l
```

List status information for all services available through the port monitor with tag ports:

```
pmadm −l −p ports
```

List the same information in condensed format:

```
pmadm −L −p ports
```

List status information for all services available through port monitors of type listen:

```
pmadm −l −t listen
```

Print the per-service configuration script associated with the service svctag available through port monitor pmtag:

```
pmadm −g −p pmtag −s svctag
```

FILES

/etc/saf/*pmtag*/_config

/etc/saf/*pmtag*/*svctag*

/var/saf/*pmtag*/*

SEE ALSO

sacadm(1M), sac(1M), doconfig(3C).

System Administrator's Reference Manual

postdaisy(1M)
Line Printer PostScript Utilities

NAME

`postdaisy` - PostScript translator for Diablo 630 files

SYNOPSIS

`/usr/lib/lp/postscript/postdaisy` [*options*] [*files*]

DESCRIPTION

`postdaisy` translates Diablo 630 daisy-wheel *files* into PostScript and writes the results to standard output. If no *files* are specified, or if – is one of the input *files*, standard input is read.

OPTIONS

–c*num*
: Prints *num* copies of each page. By default only one copy is printed.

–f*name*
: Prints text using font *name*. Any PostScript font can be used, although the best results will only be obtained with constant width fonts. The default font is Courier.

–h*num*
: Sets the initial horizontal motion index to *num*. Determines the character advance and the default point size, unless the –s option is used. The default is 12.

–m*num*
: Magnifies each logical page by the factor *num*. Pages are scaled uniformly about the origin, which by default is located at the center of each page. The default magnification is 1.0.

–n*num*
: Prints *num* logical pages on each piece of paper, where *num* can be any positive integer. By default *num* is set to 1.

–o*list*
: Prints pages whose numbers are given in the comma-separated *list*. The list contains single numbers *N* and ranges *N1* – *N2*. A missing *N1* means the lowest numbered page, a missing *N2* means the highest.

–p*mode*
: Prints *files* in either **portrait** or **landscape** *mode*. Only the first character of *mode* (`p` or `l`) is significant. The default *mode* is portrait.

–r*num*
: Selects carriage return and line feed behavior. If *num* is 1 a line feed generates a carriage return. If *num* is 2 a carriage return generates a line feed. Setting *num* to 3 enables both modes.

–s*num*
: Uses point size *num* instead of the default value set by the initial horizontal motion index.

–v*num*
: Sets the initial vertical motion index to *num*. The default is 8.

–x*num*
: Translates the origin *num* inches along the positive x axis. The default coordinate system has the origin fixed at the center of the page, with positive x to the right and positive y up the page. Positive *num* moves everything right. The default offset is 0.25 inches.

–y*num*
: Translates the origin *num* inches along the positive y axis. Positive *num* moves everything up the page. The default offset is 0.25 inches.

–A*file*

Appends a simple accounting record to *file* after all the input *files* have been successfully translated. By default no accounting data is produced.

–L*file*
Uses *file* as the PostScript prologue, which by default is `/usr/lib/lp/postscript/postdaisy.ps`.

In addition, three options allow the insertion of arbitrary PostScript at controlled points in the translation process:

–C*file*
Copies *file* to the output file. *file* follows the prologue but precedes any job initialization commands. *file* becomes part of the job's global environment and must contain legitimate PostScript commands.

–P*string*
Adds *string* to the output file. *string* follows the prologue but precedes any job initialization commands. *string* becomes part of the job's global environment and must be legitimate PostScript.

–R*action*
Requests special *action* (e.g. manual feed) on a per page or global basis. The *action* string can be given as request, request:page, or request:page:file. If page is omitted or given as 0 the request applies to all pages. If file is omitted the request lookup is done in `/usr/lib/lp/postscript/ps.requests`. The collection of recognized requests can be modified or extended by changing `/usr/lib/lp/postscript/ps.requests`. Multiple occurrences of the `–R` option behave as expected.

DIAGNOSTICS

0exit status is returned if *files* were successfully processed.

FILES

`/usr/lib/lp/postscript/postdaisy.ps`
`/usr/lib/lp/postscript/forms.ps`
`/usr/lib/lp/postscript/ps.requests`

SEE ALSO

`dpost`(1M), `postdmd`(1M), `postio`(1M), `postmd`(1M), `postprint`(1M), `postreverse`(1M), `posttek`(1M).

System Administrator's Reference Manual

postdmd(1M)
Line Printer PostScript Utilities

NAME

`postdmd -` PostScript translator for DMD bitmap files

SYNOPSIS

`/usr/lib/lp/postscript/postdmd` [*options*] [*files*]

DESCRIPTION

`postdmd` translates DMD bitmap *files*, as produced by `dmdps`, or *files* written in the Ninth Edition `bitfile(9.5)` format into PostScript and writes the results to standard output. If no *files* are specified, or if – is one of the input *files*, standard input is read.

OPTIONS

–b*num*
: Packs the bitmap in the output file using *num* byte patterns. A value of 0 turns off all packing of the output file. By default *num* is 6.

–c*num*
: Prints *num* copies of each page. By default only one copy is printed.

–f
: Flips the sense of the bits in *files* before printing the bitmaps.

–m*num*
: Magnifies each logical page by the factor *num*. Pages are scaled uniformly about the origin, which by default is located at the center of each page. The default magnification is 1.0.

–n*num*
: Prints *num* logical pages on each piece of paper, where *num* can be any positive integer. By default *num* is set to 1.

–o*list*
: Prints pages whose numbers are given in the comma-separated *list*. The list contains single numbers *N* and ranges *N1 – N2*. A missing *N1* means the lowest numbered page, a missing *N2* means the highest.

–p*mode*
: Prints *files* in either **portrait** or **landscape** *mode*. Only the first character of *mode* (`p` or `l`) is significant. The default *mode* is portrait.

–u
: Disables much of the unpacking for Eighth Edition bitmap files. Usually results in smaller output files that take longer to print. Not a recommended option.

–x*num*
: Translates the origin *num* inches along the positive x axis. The default coordinate system has the origin fixed at the center of the page, with positive x to the right and positive y up the page. Positive *num* moves everything right. The default offset is 0 inches.

–y*num*
: Translates the origin *num* inches along the positive y axis. Positive *num* moves everything up the page. The default offset is 0 inches.

–A*file*
: Appends a simple accounting record to *file* after all the input *files* have been successfully translated. By default no accounting data is produced.

–L*file*
: Uses *file* as the PostScript prologue, which by default is `/usr/lib/lp/postscript/postdmd.ps`.

In addition, three options allow the insertion of arbitrary PostScript at controlled points in the translation process:

–C*file*
: Copies *file* to the output file. *file* follows the prologue but precedes any job initialization commands. *file* becomes part of the job's global environment and must contain legitimate PostScript commands.

–P*string*
: Adds *string* to the output file. *string* follows the prologue but precedes any job initialization commands. *string* becomes part of the job's global environment and must be legitimate PostScript.

–R*action*
: Requests special *action* (e.g. manual feed) on a per page or global basis. The *action* string can be given as request, request:page, or request:page:file. If page is omitted or given as 0 the request applies to all pages. If file is omitted the request lookup is done in `/usr/lib/lp/postscript/ps.requests`. The collection of recognized requests can be modified or extended by changing `/usr/lib/lp/postscript/ps.requests`. Multiple occurrences of the `–R` option behave as expected.

Only one bitmap is printed on each logical page, and each of the input *files* must contain complete descriptions of at least one bitmap. Decreasing the pattern size using the `–b` option may help throughput on printers with fast processors (e.g. PS-810s), while increasing the pattern size will often be the right move on older models (e.g. PS-800s).

DIAGNOSTICS

0exit status is returned if *files* were successfully processed.

FILES

`/usr/lib/lp/postscript/postdmd.ps`
`/usr/lib/lp/postscript/forms.ps`
`/usr/lib/lp/postscript/ps.requests`

SEE ALSO

`dpost`(1M), `postdaisy`(1M), `postio`(1M), `postmd`(1M), `postprint`(1M), `postreverse`(1M), `posttek`(1M).

System Administrator's Reference Manual

postio(1M)
Line Printer PostScript Utilities

NAME

`postio -` serial interface for PostScript printers

SYNOPSIS

`/usr/lib/lp/postscript/postio –l`*line* [*options*] [*files*]

DESCRIPTION

`postio` sends *files* to the PostScript printer attached to *line*. If no *files* are specified, the output is sent to stdin.

OPTIONS

The first group of *options* should be sufficient for most applications:

`–b`*speed*
: Transmits data over *line* at baud rate *speed*. Recognized baud rates are 1200, 2400, 4800, 9600, and 19200. The default *speed* is 9600 baud.

`–l`*line*
: Connects to the printer attached to *line*. In most cases there is no default and `postio` must be able to read and write *line*. If the *line* doesn't begin with a `/` it may be treated as a Datakit destination.

`–q`
: Prevents status queries while *files* are being sent to the printer. When status queries are disabled a dummy message is appended to the log file before each block is transmitted.

`–B`*num*
: Sets the internal buffer size for reading and writing *files* to *num* bytes. By default *num* is 2048 bytes.

`–D`
: Enables debug mode. Guarantees that everything read on *line* will be added to the log file (standard error by default).

`–L`*file*
: Data received on *line* gets put in *file*. The default log *file* is standard error. Printer or status messages that don't indicate a change in state are not normally written to *file* but can be forced out using the `–D` option.

`–P`*string*
: Sends *string* to the printer before any of the input files. The default *string* is simple PostScript code that disables timeouts.

`–R`*num*
: Runs `postio` as a single process if *num* is 1 or as separate read and write processes if *num* is 2. By default `postio` runs as a single process.

The next two *options* are provided for users who expect to run `postio` on their own. Neither is suitable for use in spooler interface programs:

`–i`
: Runs the program in interactive mode. If *files* are available, they are sent first and output is then sent to stdin. Forces separate read and write processes and overrides many other options. To exit interactive mode use your interrupt or quit character. To get a friendly interactive connection with the printer type `executive` on a line by itself.

`–t`
: Data received on *line* and not recognized as printer or status information is written to stdout. Forces separate read and write processes. Convenient if you have a PostScript program that will be returning useful data to the host.

The last option is not generally recommended and should only be used if all else fails to provide a reliable connection:

`-S` Slows down the transmission of data to the printer. Severely limits throughput, runs as a single process, disables the `-q` option, limits the internal buffer size to 1024 bytes, can use an excessive amount of CPU time, and does nothing in interactive mode.

The best performance will usually be obtained by using a large internal buffer (the `-B` option) and by running the program as separate read and write processes (the `-R 2` option). Inability to fork the additional process causes `postio` to continue as a single read/write process. When one process is used only data sent to the printer is flow controlled.

The *options* are not all mutually exclusive. The `-i` option has priority over other options, selecting its own settings for whatever is needed to run interactive mode, independent of anything else found on the command line. Interactive mode runs as separate read and write processes and few of the other *options* accomplish anything in the presence of the `-i` option. The `-t` option needs a reliable two way connection to the printer and therefore tries to force separate read and write processes. The `-S` option relies on the status query mechanism, so `-q` is disabled and the program runs as a single process.

In most cases `postio` starts by making a connection to *line* and then attempts to force the printer into the IDLE state by sending an appropriate sequence of ^T (status query), ^C (interrupt), and ^D (end of job) characters. When the printer goes IDLE *files* are transmitted along with an occasional ^T (unless the `-q` option was used). After all the *files* are sent the program waits until it's reasonably sure the job is complete. Printer generated error messages received at any time except while establishing the initial connection (or when running interactive mode) cause `postio` to exit with a non-zero status. In addition to being added to the log file, printer error messages are also echoed to standard error.

EXAMPLES

Run as a single process at 9600 baud and send file1 and file2 to the printer attached to `/dev/tty01`:

```
postio -l /dev/tty01 file1 file2
```

Same as above except two processes are used, the internal buffer is set to 4096 bytes, and data returned by the printer gets put in file log:

```
postio -R2 -B4096 -l/dev/tty01 -Llog file1 file2
```

Establish an interactive connection with the printer at Datakit destination my/printer:

```
postio -i -l my/printer
```

Send file program to the printer connected to `/dev/tty22`, recover any data in file results, and put log messages in file log:

```
postio -t -l /dev/tty22 -L log program >results
```

WARNINGS

The input *files* are handled as a single PostScript job. Sending several different jobs, each with their own internal end of job mark (^D) is not guaranteed to work properly. `postio` may quit before all the jobs have completed and could be restarted before the last one finishes.

All the capabilities described above may not be available on every machine or even across the different versions of UNIX that are currently supported by the program. For example, the code needed to connect to a Datakit destination may only work on System V and may require that the DKHOST software package be available at compile time.

There may be no default *line* so using `-l` option is strongly recommended. If omitted `postio` may attempt to connect to the printer using standard output. If Datakit is involved the `-b` may be ineffective and attempts by `postio` to flow control data in both directions may not work. The `-q` option can help if the printer is connected to RADIAN. The `-S` option is not generally recommended and should only be used if all else fails to establish a reliable connection.

DIAGNOSTICS

A 0 exit status is returned if the files ran successfully. System errors (e.g. can't open the line) set the low order bit in the exit status, while PostScript errors set bit 1. An exit status of 2 usually means the printer detected a PostScript error in the input *files*.

SEE ALSO

dpost(1M), postdaisy(1M), postdmd(1M), postmd(1M), postprint(1M), postreverse(1M), posttek(1M).

System Administrator's Reference Manual

postmd(1M)
Line Printer PostScript Utilities

NAME

`postmd` - matrix display program for PostScript printers

SYNOPSIS

`/usr/lib/lp/postscript/postmd` [*options*] [*files*]

DESCRIPTION

`postmd` reads a series of floating point numbers from *files*, translates them into a PostScript gray scale image, and writes the results to standard output. In a typical application the numbers might be the elements of a large matrix, written in row major order, while the printed image could help locate patterns in the matrix. If no *files* are specified, or if – is one of the input *files*, standard input is read.

OPTIONS

–b*num*
: Packs the bitmap in the output file using *num* byte patterns. A value of 0 turns off all packing of the output file. By default *num* is 6.

–c*num*
: Prints *num* copies of each page. By default only one copy is printed.

–d*dimen*
: Sets the default matrix dimensions for all input *files* to *dimen*. The *dimen* string can be given as rows or rows`x`columns. If columns is omitted it will be set to rows. By default `postmd` assumes each matrix is square and sets the number of rows and columns to the square root of the number of elements in each input file.

–g*list*
: *list* is a comma or space separated string of integers, each lying between 0 and 255 inclusive, that assigns PostScript gray scales to the regions of the real line selected by the `–i` option. 255 corresponds to white and 0 to black. `postmd` assigns a default gray scale that omits white (i.e. 255) and gets darker as the regions move from left to right along the real line.

–i*list*
: *list* is a comma or space separated string of N floating point numbers that partition the real line into 2N+1 regions. The *list* must be given in increasing numerical order. The partitions are used to map floating point numbers read from the input *files* into gray scale integers that are assigned automatically by `postmd` or arbitrarily selected using the `–g` option. The default interval *list* is "-1,0,1" which partions the real line into 7 regions.

–m*num*
: Magnifies each logical page by the factor *num*. Pages are scaled uniformly about the origin, which by default is located at the center of each page. The default magnification is 1.0.

–n*num*
: Prints *num* logical pages on each piece of paper, where *num* can be any positive integer. By default *num* is set to 1.

–o*list*
: Prints pages whose numbers are given in the comma-separated *list*. The list contains single numbers *N* and ranges *N1 – N2*. A missing *N1* means the lowest numbered page, a missing *N2* means the highest.

–p*mode*
: Prints *files* in either **portrait** or **landscape** *mode*. Only the first character of *mode* (`p` or `l`) is significant. The default *mode* is portrait.

–w*window*

window is a comma or space separated list of four positive integers that select the upper left and lower right corners of a submatrix from each of the input *files*. Row and column indices start at 1 in the upper left corner and the numbers in the input *files* are assumed to be written in row major order. By default the entire matrix is displayed.

−x*num*
: Translates the origin *num* inches along the positive x axis. The default coordinate system has the origin fixed at the center of the page, with positive x to the right and positive y up the page. Positive *num* moves everything right. The default offset is 0 inches.

−y*num*
: Translates the origin *num* inches along the positive y axis. Positive *num* moves everything up the page. The default offset is 0 inches.

−A*file*
: Appends a simple accounting record to *file* after all the input *files* have been successfully translated. By default no accounting data is produced.

−L*file*
: Uses *file* as the PostScript prologue, which by default is `/usr/lib/lp/postscript/postmd.ps`.

In addition, three options allow the insertion of arbitrary PostScript at controlled points in the translation process:

−C*file*
: Copies *file* to the output file. *file* follows the prologue but precedes any job initialization commands. *file* becomes part of the job's global environment and must contain legitimate PostScript commands.

−P*string*
: Adds *string* to the output file. *string* follows the prologue but precedes any job initialization commands. *string* becomes part of the job's global environment and must be legitimate PostScript.

−R*action*
: Requests special *action* (e.g. manual feed) on a per page or global basis. The *action* string can be given as request, request:page, or request:page:file. If page is omitted or given as 0 the request applies to all pages. If file is omitted the request lookup is done in `/usr/lib/lp/postscript/ps.requests`. The collection of recognized requests can be modified or extended by changing `/usr/lib/lp/postscript/ps.requests`. Multiple occurrences of the `−R` option behave as expected.

Only one matrix is displayed on each logical page, and each of the input *files* must contain complete descriptions of exactly one matrix. Matrix elements are floating point numbers arranged in row major order in each input file. White space, including newlines, is not used to determine matrix dimensions. By default `postmd` assumes each matrix is square and sets the number of rows and columns to the square root of the number of elements in the input file. Supplying default dimensions on the command line using the `−d` option overrides this default behavior, and in that case the dimensions apply to all input *files*.

An optional header can be supplied with each input file, which sets the matrix dimensions, the partition of the real line, the gray scale map, and a window into the matrix. The header consists of keyword/value pairs, each on a separate line. It begins on the first line of each input file and ends with the first unrecognized string, i.e. the first matrix element. Values set in the header take precedence, but only apply to the current input file. Recognized header keywords are `dimension`, `interval`, `grayscale`, and `window`. The syntax of the value string that follows each keyword parallels what's accepted by the `−d`, `−i`, `−g`, and `−w` options.

EXAMPLES

For example, suppose file initially contains the 1000 numbers in a 20x50 matrix. Then the command line,

```
postmd -d20x50 -i"-100 100" -g0,128,254,128,0 file
```

and prepending the header,

```
dimension 20x50
interval  -100.0 .100e+3
```

```
grayscale 0 128 254 128 0
```

to file and typing the command line,

```
postmd file
```

produce exactly the same output. The interval list partitions the real line into five regions and the gray scale list maps numbers less than −100 or greater than 100 into 0 (i.e. black), numbers equal to −100 or 100 into 128 (i.e. 50 percent black), and numbers between −100 and 100 into 254 (i.e. almost white).

WARNINGS

The largest matrix that can be adequately displayed is a function of the interval and gray scale lists, the printer resolution, and the paper size. A 600x600 matrix is an optimistic upper bound for a two element interval list (i.e. five regions) using 8.5x11 inch paper on a 300 dpi printer.

Using white (i.e. 255) in a gray scale list is not recommended and won't show up in the legend and bar graph that `postmd` displays below each image.

DIAGNOSTICS

0exit status is returned if *files* were successfully processed.

FILES

```
/usr/lib/lp/postscript/postmd.ps
/usr/lib/lp/postscript/forms.ps
/usr/lib/lp/postscript/ps.requests
```

SEE ALSO

`dpost`(1M), `postdaisy`(1M), `postdmd`(1M), `postio`(1M), `postprint`(1M), `postreverse`(1M), `posttek`(1M).

System Administrator's Reference Manual

postplot(1M)
Line Printer PostScript Utilities

NAME

`postplot` - PostScript translator for plot graphics files

SYNOPSIS

`/usr/lib/lp/postscript/postplot` [*options*] [*files*]

DESCRIPTION

`postplot` translates `plot` graphics *files* into PostScript and writes the results to standard output. If no *files* are specified, or if – is one of the input *files*, standard input is read.

OPTIONS

–c*num*
: Prints *num* copies of each page. By default only one copy is printed.

–f*name*
: Prints text using font *name*. Any PostScript font can be used, although the best results will only be obtained with constant width fonts. The default font is Courier.

–m*num*
: Magnifies each logical page by the factor *num*. Pages are scaled uniformly about the origin, which by default is located at the center of each page. The default magnification is 1.0.

–n*num*
: Prints *num* logical pages on each piece of paper, where *num* can be any positive integer. By default *num* is set to 1.

–o*list*
: Prints pages whose numbers are given in the comma-separated *list*. The list contains single numbers *N* and ranges *N1 – N2*. A missing *N1* means the lowest numbered page, a missing *N2* means the highest.

–p*mode*
: Prints *files* in either **portrait** or **landscape** *mode*. Only the first character of *mode* (`p` or `l`) is significant. The default *mode* is portrait.

–w*num*
: Sets the line width used for graphics to *num* points, where a point is approximately 1/72 of an inch. By default *num* is set to 0 points, which forces lines to be one pixel wide.

–x*num*
: Translates the origin *num* inches along the positive x axis. The default coordinate system has the origin fixed at the center of the page, with positive x to the right and positive y up the page. Positive *num* moves everything right. The default offset is 0 inches.

–y*num*
: Translates the origin *num* inches along the positive y axis. Positive *num* moves everything up the page. The default offset is 0 inches.

–A*file*
: Appends a simple accounting record to *file* after all the input *files* have been successfully translated. By default no accounting data is produced.

–L*file*
: Uses *file* as the PostScript prologue, which by default is `/usr/lib/lp/postscript/postplot.ps`.

In addition, three options allow the insertion of arbitrary PostScript at controlled points in the translation process:

–C*file*

Copies *file* to the output file. *file* follows the prologue but precedes any job initialization commands. *file* becomes part of the job's global environment and must contain legitimate PostScript commands.

−P*string*
Adds *string* to the output file. *string* follows the prologue but precedes any job initialization commands. *string* becomes part of the job's global environment and must be legitimate PostScript.

−R*action*
Requests special *action* (e.g. manual feed) on a per page or global basis. The *action* string can be given as request, request:page, or request:page:file. If page is omitted or given as 0 the request applies to all pages. If file is omitted the request lookup is done in `/usr/lib/lp/postscript/ps.requests`. The collection of recognized requests can be modified or extended by changing `/usr/lib/lp/postscript/ps.requests`. Multiple occurrences of the `−R` option behave as expected.

DIAGNOSTICS

0exit status is returned if *files* were successfully processed.

NOTES

The default line width is too small for write-white print engines, like the one used by the PS-2400.

FILES

`/usr/lib/lp/postscript/postplot.ps`
`/usr/lib/lp/postscript/forms.ps`
`/usr/lib/lp/postscript/ps.requests`

SEE ALSO

`dpost`(1M), `postdaisy`(1M), `postdmd`(1M), `postio`(1M), `postmd`(1M), `postprint`(1M), `postreverse`(1M).

System Administrator's Reference Manual

postprint(1M)
Line Printer PostScript Utilities

NAME

`postprint -` PostScript translator for text files

SYNOPSIS

`/usr/lib/lp/postscript/postprint` [*options*] [*files*]

DESCRIPTION

`postprint` translates text *files* into PostScript and writes the results to standard output. If no *files* are specified, or if – is one of the input *files*, standard input is read.

OPTIONS

–c*num*
: Prints *num* copies of each page. By default only one copy is printed.

–e
: Outputs octal escapes for non-printing ASCII characters.

–f*name*
: Prints text using font *name*. Any PostScript font can be used, although the best results will only be obtained with constant width fonts. The default font is Courier.

–l*num*
: Sets the length of a page to *num* lines. By default *num* is 66. Setting *num* to 0 is allowed, and will cause `postprint` to guess a value, based on the point size that's being used.

–m*num*
: Magnifies each logical page by the factor *num*. Pages are scaled uniformly about the origin, which by default is located at the center of each page. The default magnification is 1.0.

–n*num*
: Prints *num* logical pages on each piece of paper, where *num* can be any positive integer. By default *num* is set to 1.

–o*list*
: Prints pages whose numbers are given in the comma-separated *list*. The list contains single numbers *N* and ranges *N1* – *N2*. A missing *N1* means the lowest numbered page, a missing *N2* means the highest.

–p*mode*
: Prints *files* in either **portrait** or **landscape** *mode*. Only the first character of *mode* (`p` or `l`) is significant. The default *mode* is portrait.

–r*num*
: Selects carriage return behavior. Carriage returns are ignored if *num* is 0, cause a return to column 1 if *num* is 1, and generate a newline if *num* is 2. The default *num* is 0.

–s*num*
: Prints *files* using point size *num*. When printing in landscape mode *num* is scaled by a factor that depends on the imaging area of the device. The default size for portrait mode is 10.

–t*num*
: Assumes tabs are set every *num* columns, starting with the first column. By default tabs are set every 8 columns.

–x*num*
: Translates the origin *num* inches along the positive x axis. The default coordinate system has the origin fixed at the center of the page, with positive x to the right and positive y up the page. Positive *num* moves everything right. The default offset is 0.25 inches.

–y*num*

Translates the origin *num* inches along the positive y axis. Positive *num* moves everything up the page. The default offset is 0.25 inches.

−A*file*
Appends a simple accounting record to *file* after all the input *files* have been successfully translated. By default no accounting data is produced.

−L*file*
Uses *file* as the PostScript prologue, which by default is `/usr/lib/lp/postscript/postprint.ps`.

In addition, three options allow the insertion of arbitrary PostScript at controlled points in the translation process:

−C*file*
Copies *file* to the output file. *file* follows the prologue but precedes any job initialization commands. *file* becomes part of the job's global environment and must contain legitimate PostScript commands.

−P*string*
Adds *string* to the output file. *string* follows the prologue but precedes any job initialization commands. *string* becomes part of the job's global environment and must be legitimate PostScript.

−R*action*
Requests special *action* (e.g. manual feed) on a per page or global basis. The *action* string can be given as request, request:page, or request:page:file. If page is omitted or given as 0 the request applies to all pages. If file is omitted the request lookup is done in `/usr/lib/lp/postscript/ps.requests`. The collection of recognized requests can be modified or extended by changing `/usr/lib/lp/postscript/ps.requests`. Multiple occurrences of the `−R` option behave as expected.

A new logical page is started after 66 lines have been printed on the current page, or whenever an ASCII form feed character is read. The number of lines per page can be changed using the `−l` option. Unprintable ASCII characters are ignored, and lines that are too long are silently truncated by the printer.

EXAMPLES

Print file1 and file2 in landscape mode,

```
postprint -pland file1 file2
```

Print 3 logical pages on each physical page in portrait mode,

```
postprint -n3 file
```

DIAGNOSTICS

0exit status is returned if *files* were successfully processed.

FILES

```
/usr/lib/lp/postscript/postprint.ps
/usr/lib/lp/postscript/forms.ps
/usr/lib/lp/postscript/ps.requests
```

SEE ALSO

dpost(1M), postdaisy(1M), postdmd(1M), postio(1M), postmd(1M), postreverse(1M), posttek(1M).

System Administrator's Reference Manual

postreverse(1M)
Line Printer PostScript Utilities

NAME

`postreverse -` reverse the page order in a PostScript file

SYNOPSIS

`/usr/lib/lp/postscript/postreverse` [*options*] [*files*]

DESCRIPTION

`postreverse` reverses the page order in files that conform to Adobe's Version 1.0 or Version 2.0 file structuring conventions, and writes the results to standard output. Only one input *file* is allowed and if no *file* is specified, standard input is read.

OPTIONS

−o*list*
: Selects pages whose numbers are given in the comma-separated *list*. The list contains single numbers *N* and ranges *N1* − *N2*. A missing *N1* means the lowest numbered page, a missing *N2* means the highest.

`−r` Don't reverse the pages in *file*.

−T*dir*
: Uses *dir* as the temporary file directory when reading from standard input. By default *dir* is set to `/tmp`.

`postreverse` can handle a limited class of files that violate page independence, provided all global definitions are bracketed by `%%BeginGlobal` and `%%EndGlobal` comments. In addition, files that mark the end of each page with `%%EndPage: label ordinal` comments will also reverse properly, provided the prologue and trailer sections can be located. If *postreverse* fails to find an `%%EndProlog` or `%%EndSetup` comment the entire *file* is copied, unmodified, to standard output.

Since global definitions are pulled out of individual pages and put in the prologue, the output file can be minimally conforming, even if the input *file* wasn't.

EXAMPLES

Select pages 1 to 100 from file and reverse the pages,

```
postreverse -o1-100 file
```

Print 4 logical pages on each physical page and reverse all the pages,

```
postprint -n4 file | postreverse
```

Produce a minimally conforming file from output generated by dpost without reversing the pages,

```
dpost file | postreverse -r
```

DIAGNOSTICS

0exit status is returned if *files* were successfully processed.

NOTES

No attempt has been made to deal with redefinitions of global variables or procedures. If standard input is used, the input *file* will be read three times before being reversed.

SEE ALSO

`dpost`(1M), `postdaisy`(1M), `postdmd`(1M), `postio`(1M), `postmd`(1M), `postprint`(1M), `posttek`(1M).

System Administrator's Reference Manual

posttek(1M)
Line Printer PostScript Utilities

NAME

`posttek` - PostScript translator for tektronix 4014 files

SYNOPSIS

`/usr/lib/lp/postscript/posttek` [*options*] [*files*]

DESCRIPTION

`posttek` translates tektronix 4014 graphics *files* into PostScript and writes the results to standard output. If no *files* are specified, or if – is one of the input *files*, standard input is read.

OPTIONS

–c*num*
: Prints *num* copies of each page. By default only one copy is printed.

–f*name*
: Prints text using font *name*. Any PostScript font can be used, although the best results will only be obtained with constant width fonts. The default font is Courier.

–m*num*
: Magnifies each logical page by the factor *num*. Pages are scaled uniformly about the origin, which by default is located at the center of each page. The default magnification is 1.0.

–n*num*
: Prints *num* logical pages on each piece of paper, where *num* can be any positive integer. By default *num* is set to 1.

–o*list*
: Prints pages whose numbers are given in the comma-separated *list*. The list contains single numbers *N* and ranges *N1 – N2*. A missing *N1* means the lowest numbered page, a missing *N2* means the highest.

–p*mode*
: Prints *files* in either **portrait** or **landscape** *mode*. Only the first character of *mode* (`p` or `l`) is significant. The default *mode* is portrait.

–w*num*
: Sets the line width used for graphics to *num* points, where a point is approximately 1/72 of an inch. By default *num* is set to 0 points, which forces lines to be one pixel wide.

–x*num*
: Translates the origin *num* inches along the positive x axis. The default coordinate system has the origin fixed at the center of the page, with positive x to the right and positive y up the page. Positive *num* moves everything right. The default offset is 0 inches.

–y*num*
: Translates the origin *num* inches along the positive y axis. Positive *num* moves everything up the page. The default offset is 0 inches.

–A*file*
: Appends a simple accounting record to *file* after all the input *files* have been successfully translated. By default no accounting data is produced.

–L*file*
: Uses *file* as the PostScript prologue, which by default is `/usr/lib/lp/postscript/posttek.ps`.

In addition, three options allow the insertion of arbitrary PostScript at controlled points in the translation process:

–C*file*

Copies *file* to the output file. *file* follows the prologue but precedes any job initialization commands. *file* becomes part of the job's global environment and must contain legitimate PostScript commands.

−P*string*
Adds *string* to the output file. *string* follows the prologue but precedes any job initialization commands. *string* becomes part of the job's global environment and must be legitimate PostScript.

−R*action*
Requests special *action* (e.g. manual feed) on a per page or global basis. The *action* string can be given as request, request:page, or request:page:file. If page is omitted or given as 0 the request applies to all pages. If file is omitted the request lookup is done in `/usr/lib/lp/postscript/ps.requests`. The collection of recognized requests can be modified or extended by changing `/usr/lib/lp/postscript/ps.requests`. Multiple occurrences of the `−R` option behave as expected.

DIAGNOSTICS

0exit status is returned if *files* were successfully processed.

NOTES

The default line width is too small for write-white print engines, like the one used by the PS-2400.

FILES

`/usr/lib/lp/postscript/posttek.ps`
`/usr/lib/lp/postscript/forms.ps`
`/usr/lib/lp/postscript/ps.requests`

SEE ALSO

`dpost`(1M), `postdaisy`(1M), `postdmd`(1M), `postio`(1M), `postmd`(1M), `postprint`(1M), `postreverse`(1M).

System Administrator's Reference Manual

powerdown(1M)

NAME

`powerdown -` power down the system and change the system status.

SYNOPSIS

`powerdown`

DESCRIPTION

You can change the status of a computer using the `powerdown` command which is also available as a login with the same name. In general it is used to switch from multiuser status to another status.

The command calls a visual interface. An identical function is also available for this purpose in the `sysadm` menu, i.e.:

`sysadm shutdown`

DIAGNOSTICS

The `powerdown` command has the following return codes.

`0` = Task completed successfully.

`2` = Bad command syntax. A message with the correct syntax is displayed.

`6` = No entry in `/etc/passwd`. The "powerdown" user must be configured.

`7` = There is no visual interface available for this command as `fmli` cannot be called. (The FMLI package is either not installed or damaged.)

SEE ALSO

`shutdown`(1M), `sysadm`(1M).

System Administrator's Reference Manual

proceso(1M)

NAME

proceso - list processes and kernel memory usage

SYNOPSIS

proceso [–s] [–p[v[v]] [–k] [–M]

proceso –c[v[v]]

proceso –r

proceso –u[u]]

ps | proceso [–s] [–p[v[v]]] [–k] [–M]

DESCRIPTION

proceso reads the processes and kernel segments page tables. These page tables hold the page table entries for each page in the address space of processes and the kernel. By checking the individual flags of the page table entries, proceso can determine which pages are resident in the main memory and whether the pages are private to that process or shared with other processes. System processes cannot be analyzed.

The user's access permissions determine how the individual processes can be accessed.

OPTIONS

–s Summary information about memory usage.

–p Summary information about all selected processes.

–pv Detailed information about specified processes.

–pvv Detailed information about specified processes including their segments.

–k Lists detailed information about each kernel segment.

–M Physical memory usage per process.

–c Displays running and ready-to-run processes.

–cv Extended display of running and ready-to-run processes.

–cvv Detailed display of running and ready-to-run processes.

–u Scans the reference file for memory mapped files proceso.ref and update the device and inode entries. Use this option if file systems were mounted differently or if you use a proceso.ref file from another machine.

–uu Scans all mounted file systems for all entries that have not yet been assigned in proceso.ref.

Note: this may take a long time.

–r Scans all directories listed in the proceso.dirs file and adds entries to proceso.ref file.

PROCESS MEMORY TYPES

When the –p option is used, proceso lists the memory usage for each process selected. The address space may consist of three sections:

process private

This type of memory is used solely by that process. This is typically initialized data, memory allocated using malloc(3C) , and the stack segment.

shared files

Reliant UNIX supports the concept of memory mapped files. This is a mechanism by which the data of a file is mapped into the address space of a process. This means a process can access this data by using an address operation. See also mmap(2) . Examples of such memory mapped files are the code

segments of an executable or a library.

shared memory
Shared memory is a mechanism of the interprocess communication facilities (IPC) of Reliant UNIX. Using shared memory processes you can communicate and exchange data very quickly.

SUMMARY

If no option is specified, `proceso` will give a summary of the system's memory usage. Please note that in all following summaries `core` stands for the size of main memory used and `total` designates the virtual size. 1 KB is equal to 1024 bytes. When specifying the `-p` option, `proceso` will also analyze the processes memory usage.

```
******* SUMMARY OF SYSTEM MEMORY USAGE *******
type             |number | core(KB)  |total(KB)
-----------------+-------+----------+-------------
process private  |   438 |    74044 |  132964
mapped files     |   112 |    18912 |   24132
kernel memory    |       |    74384 |    186636
memory disk      |     1 |    65536 |
shared memory    |     0 |        0 |       0
not ref. pages   |       |       48 |
kernel size      |       |    15695 |
free memory      |       |    56256 |
-----------------+-------+----------+-------------
proceso
TOTAL    |       |   304875 |
-----------------+-------+----------+-------------
physical memory  |       |   327680 |
```

Note:

The final total may differ by up to 10 percent from the physical memory size available in the system.

PROCESS MEMORY ANALYSIS

Using the `-pv` option gives a more detailed summary for each process and each memory mapped file in the system.

```
******* ANALYSIS OF SYSTEM MEMORY USAGE BY PROCESSES *******
       |process private| shared files | shared memory |
  pid |  core   total | core   total |  core   total | cmd
------+---------------+--------------+---------------+-----------------------
  661 |     44    128 |  268     720 |     0       0 | /usr/lib/saf/sac -t 300
 9260 |    132    484 |  396     408 |     0       0 | vi xioinfo.h xiospb.h
  267 |      4     24 |   28      68 |     0       0 | cat /dev/osm
  324 |     28     72 |
204     428 |     0       0 | /opt/log3/bin/log3svr -etsdm1440
  325 |     28     72 |  152     412 |     0       0 | /opt/log3/bin/log3tlog2
  326 |     36     64 |  168     420 |     0       0 | /opt/log3/bin/log3syslogd
  490 |      0    324 |    0     716 |     0       0 | /usr/lib/netsvc/rwall/rpc.
rwalld
  158 |    436    636 |  292     772 |     0       0 | /sbin/errord
  454 |    240    480 |  416     812 |     0       0 | /usr/sbin/rpcbind
  298 |      8     52 |  164     412 |     0       0 | /usr/sbin/silsd

349 |    124    340 |  320     824 |     0       0 | /usr/sbin/syslogd
   :
   :
 6852 |    108    192 |  744     820 |     0       0 | in.rlogind
```

```
 9950 |    356    492 | 3524    4144 |     0       0 | mwm□
14415 |     36     48 |  168     180 |     0       0 | /usr/lib/X11/xdm/Xsession□
------+--------------+--------------+---------------+----------------------□
total    85832  34356 KB□
******** LIST OF MEMORY MAPPED FILES ********□
                   |  core    total |□
dev(dec) ino(dec)  |
(KB)     (KB)   | name □
-------------------+---------------+------------------------ □
 1048576     6185 |     28      68 | /sbin/cat□
 1048576     6185 |     28      68 | /usr/bin/cat□
 1048576     6217 |     24      28 | /sbin/errord□
 1048576     6240 |    168     180 | /sbin/jsh□
 1048578     4183 |    576     632 | /usr/lib/libX11.so.5.0□
   :□
 1048578     4184 |    224     284 | /usr/lib/libXaw.so□
 1048578     4184 |    224     284 | /usr/lib/libXaw.so.5.0□
 1048578     4186 |     32      40 | /usr/lib/libXext.so□
 1048578    28787 |
  280     280 | -□
-------------------+---------------+------------------------ □
total                20632   23984 KB
```

Note:

Files are only listed in the table with their full name if this is contained in the reference table `proceso.ref`. The reference table can be completed with the names using the `–uu` option.

DETAILED PROCESS ANALYSIS

Using the `–pvv` option, `proceso` gives a detailed list of all memory segments of the specified processes. `type` specifies the individual segment type which can either be a filename (memory mapped) or one of `PRIVATE`, `DATA SEGMENT` (`.data` section), `STACK` or `SHARED MEMORY`. When summing up the values of the individual segments, the segment types `PRIVATE`, `DATA SEGMENT` and `STACK` are added to the private `private` section. Segments of "Memory Mapped" files can be recognized by their filenames: "–", when the `proceso.ref` entry is missing, and are added to the value `mapped`. The `flags` field specifies the attribute flags: □

`R`
: readable

`W`
: writeable

`X`
: segment is executable

`C`
: shared segment (shared memory)

`B`
: data segment (`.data` section)

`S`
: stack segment .RE □

```
PID:  1578   CMD: //usr/lib/saf/listen tcp□
 N | start(X) | core(KB) |total(KB) |    flags     | type□
---+----------+----------+----------+--------------+-------------□
 0 |        0 |        0 |       16 |  R . X . . . | PRIVATE□
 1 |   400000 |       80 |       80 |  R . X . . . | /usr/lib/saf/listen□
 2 |   450000 |       32 |       32 |  R W X . . . | PRIVATE□
 3 |   458000 |      176 |      176 |  R W X . B . | DATA SEGMENT□
 4 |  8000000 |      320 |      480 |  R . X . . . | /usr/lib/libc.so.1□
 5 |
807C000 |       16 |       16 |  R W . . . . | PRIVATE□
```

```
 6 |  80B4000 |       48 |       48 |  R W X . . . | PRIVATE□
 7 |  80C0000 |        0 |       16 |  R W X . . . | PRIVATE□
 8 |  8100000 |      176 |      272 |  R . X . . . | /usr/lib/libnsl.so□
 9 |  8180000 |       32 |       32 |  R W X . . . | PRIVATE□
10 |  81C0000 |       96 |       96 |  R . X . . . | /usr/lib/libsocket.so□
11 |  8214000 |       16 |       16 |  R W X . . . | PRIVATE□
12 | 7FFF4000 |       32 |       32 |  R W X . . S | STACK□
---+----------+----------+----------+--------------+--------------□
private
      |      352 |      384 |□
mapped        |      672 |      928 |□
shared mem    |        0 |        0 |□
--------------+----------+----------+□
PID:  4339   CMD: /sbin/quickdd if=/dev/zero of=/dev/null bs=512 count=10000
00□
 N | start(X) | core(KB) |total(KB) |    flags     | type□
---+----------+----------+----------+--------------+--------------□
 0 |        0 |        0 |       16 |  R . X . . . | PRIVATE□
 1 |   400000 |       64 |       80 |  R . X . . . | /sbin/quickdd□
 2 |   450000 |
 16 |       16 |  R W X . . . | PRIVATE□
 3 |   454000 |        0 |       16 |  R W X . B . | DATA SEGMENT□
 4 |  8004000 |        0 |     1056 |  R W X C . . | SHARED MEMORY□
 5 |  8110000 |       16 |       16 |  R W X C . . | SHARED MEMORY□
 6 | 7FFF4000 |       16 |       32 |  R W X . . S | STACK□
---+----------+----------+----------+--------------+--------------□
private       |       32 |       80 |□
mapped        |       64 |       80 |□
shared mem    |       16 |     1072 |□
--------------+----------+----------+
```

KERNEL MEMORY USAGE

The –k option gives a detailed list of the allocated kernel segments and available (free) memory.

```
****** ANALYSIS OF SYSTEM KERNEL MEMORY USAGE *******□
start(X)         end(X)           segment           core(KB) total(KB)□
----------------+----------------+-----------------+--------+---------
                                  □
                                    SEGK                28968     36864□
                                    SEGMAP              21524     96524□
                                    SEGU                 5300     53248□
A800000001400000 A800000001800000 cache_rpl_buffer     4096□
FFFFFFFF81800000 FFFFFFFF81900000 alt_stack_area       1024□
A800000001904000 A800000001908000 cpuprivvir             16□
A80000000190C000 A800000001914000 kern_pte               32□
A800000001914000
A800000001915EE0 segmap_pte              8□
A800000001918000 A800000001924000 segu_pte               48□
A800000001924000 A800000001984000 kern_pce              384□
A800000001984000 A800000001988000 page_hash              16□
A800000001988000 A80000000198C000 page_hash_cor          16□
A80000000198C000 A800000001990000 anon_hash              16□
```

```
A800000001990000 A800000001998000 page_base_hash         32
A800000001998000 A800000001E1F800 pages                4638
                 other physical page locks by kernel    8266
-------------------------------------------------+--------+---------
KERNEL                                                74384    186636
```

PHYSICAL MEMORY USAGE

`proceso` displays the physical memory usage in relation to specific processes. The appropriate memory areas (e.g. CPU boards) are output for the individual physical addresses based on the information contained in the `proceso.sect` file.

Furthermore, the information is output for each process in accordance with **DETAILED PROCESS ANALYSIS**.

```
PID: 27709   CMD: -ksh
******** LIST OF PHYSICAL ADDRESSES ********
 S  vaddr(X) |   paddr(X):sect   paddr(X):sect   paddr(X):sect   paddr(X):sect
-------------+-------------------------------------------------------------
*         0 |   2BA80000:b21
*    400000 |   3B0C0000:b24    3D3D0000:        A100000:       2A870000:b21
     410000 |   3DB00000:        C810000:       32340000:b22    225F0000:b21
     420000 |   16100000:b16    30BB0000:b22    3CB80000:       31C90000:b22
     430000 |   139A0000:b15    1E9D0000:b17    3DA60000:        8510000:
     440000 |   1BE60000:b17
*    480000 |   3C9C0000:       18510000:b17
    D9A0000:
*    48C000 |    6590000:b14    39300000:b24    35690000:b23    1A640000:b17
*  7FFE8000 |           -       31270000:b22
 N | start(X) | core(KB) | total(KB) |    flags     | type
---+----------+----------+-----------+--------------+----------------------
 0 |        0 |       16 |        16 |  R . X . . . | PRIVATE
...
```

PROCESS INFORMATION

`proceso` displays process information which is comparable with the output of the `ps` command.

If no other parameters are specified, only those processes belonging to the current session are displayed.

The additional `v` parameter displays all processes for the user currently logged on.

If the additional `vv` parameter is specified, more detailed information is output for these processes.

All output types contain information on the processor affinity of the processes. A low `AFFSEC` value means that the process has only recently started running on this CPU. The `CACHE` value specifies how the cache is configured (number of "cache lines") by the process – this entry is only relevant for 64-bit systems.

```
******* RUNNING/RUNNABLE PROCESSES *******
  PID  PPID  UID CPU S PRI CLS AFFSEC    CACHE   USR   SYS  SLEEP CMD
15387  9754  215   1 O  34  TS   0.00  4563340  0:00  0:00   0:00 ./proceso -cvv
15388  9754  215   0 R  34  TS   0.04  4563340  0:00  0:00   0:00 pg
 9754  7464  215   1 S  34  TS   0.01  4563340  0:00  0:00   0:00 ksh
```

WORKING SET OF A PROCESS

`proceso` is very useful in determining the working set of a process and for calculating the main memory used for a specified number of users. You should use the following syntax to determine, for example, the working set of one user:

```
ps -u myuser | proceso -pv
```

Note:

`proceso` handles the first column of the `ps` output as the process ID. For this reason, only `ps` options that output the process ID in the first column should be used.

The private total process gives the memory usage for one user. You may multiply this by the number of users that use the same process environment.

NOTES

`proceso` may yield incorrect results or even core dump if the system is not booted from `/unix`. So please verify that `/unix` is linked to the kernel image that has been booted.

FILES

`/opt/bin/proceso/proceso.ref`

As `proceso` cannot find out the file name of a memory mapped file during runtime, it uses a cross-reference list containing the file name and the device and inode number. The `proceso.ref` file holds all filenames of memory mapped files and their associated device and inode numbers.

When `proceso -pv` is executed, an entry with the device and inode number is added to the file `proceso.ref` for files still without cross references. The relevant filename is completed in `proceso.ref` with `proceso -u`.

`/opt/bin/proceso/proceso.dirs`

If the `proceso.ref` file was removed or cleared for some reason, you can create a new reference file based on the directories listed in `proceso.dirs`. This file should contain the directory names of all binaries and libraries that may be used by the running processes. You may also add directory entries for specific add-on products (databases, etc.). See also the `-r` option.

`/opt/bin/proceso/proceso.sect`

The file is only relevant for the `M` function. For selected physical memory areas, (e.g. CPU board memory), it contains

1) the Id for this area (max. 4 characters) (1st column);
2) its hexadecimal start address (2nd column);
3) its hexadecimal length (3rd column).

If "0" is entered as the start address for subsequent entries, the start address is the "end address+1" of the previous entry. If the `proceso.sect` file is in the current directory it will be used.

Information on how the memory is assigned to the different boards can be displayed with `showconf -M`.

SEE ALSO

`ps(1)`, `crash(1M)`, `showconf(8)`.

System Administrator's Reference Manual

profiler(1M)

NAME

profiler:
prfld, prfstat, prfdc, prfsnap, prfpr - system profiler

SYNOPSIS

/usr/sbin/prfld [*system_namelist*]

/usr/sbin/prfstat on

/usr/sbin/prfstat off

/usr/sbin/prfdc *file* [*period* [*off_hour*]]

/usr/sbin/prfsnap *file*

/usr/sbin/prfpr *file* [*cutoff* [*system_namelist*]]

DESCRIPTION

prfld, prfstat, prfdc, prfsnap, and prfpr form a system of programs to facilitate an activity study of the Reliant UNIX operating system.

prfld is used to initialize the recording mechanism in the system. It generates a table containing the starting address of each system subroutine as extracted from *system_namelist*.

prfstat is used to enable or disable the sampling mechanism. Profiler overhead is less than 1% as calculated for 500 text addresses. prfstat will also reveal the number of text addresses being measured.

prfdc and prfsnap perform the data collection function of the profiler by copying the current value of all the text address counters to a file where the data can be analyzed. prfdc will store the counters into *file* every *period* minutes and will turn off at *off_hour* (valid values for *off_hour* are 0–24). prfsnap collects data at the time of invocation only, appending the counter values to *file*.

prfpr formats the data collected by prfdc or prfsnap. Each text address is converted to the nearest text symbol (as found in *system_namelist*) and is printed if the percent activity for that range is greater than *cutoff*. prfpr produces no output, if *file* contains only one sample. Call prfsnap twice before calling prfpr.

FILES

/dev/prf
: interface to profile data and text addresses

/stand/unix
: default for system namelist file

SEE ALSO

System Administrator's Guide.

System Administrator's Reference Manual

ptx(1M)

NAME

`ptx` - make permuted index

SYNOPSIS

`ptx` [*option* ...] [*input* [*output*]]

DESCRIPTION

`ptx` generates the file *output* that can be processed with a text formatter (`nroff` or `troff`) to produce a permuted index of file *input*. Standard input (`–`) and standard output are default.

`ptx` has three phases: the first does the permutation, generating one line for each keyword in an input line. The keyword is rotated to the front. The permuted file is then sorted. Finally, the sorted lines are rotated, so the keyword comes at the middle of each line.

`ptx` output is in the following form:

.xx "" "*before keyword*" "*keyword*" "*after keyword*"

where `.xx` is assumed to be an `nroff(1M)` or `troff(1M)` macro provided by the user, or provided by `ptx(1M)`. The `–mptx` macro package provides the `.xx` macro definition. The *before keyword* and *after keyword* fields incorporate as much of the line as will fit around the *keyword* when it is printed.

The first field and the last field, at least one of which is always the empty string, are wrapped-around pieces small enough to fit in the unused space at the opposite end of the line.

OPTIONS

`–f` Folds uppercase and lowercase letters for sorting.

`–t` Prepares the output for the phototypesetter.

`–w` *n* Uses the next argument, *n*, as the length of the output line. The default line length is 72 characters for `nroff` and 100 for `troff`.

`–g` *n* Uses the next argument, `n`, as the number of characters that `ptx` will reserve in its calculations for each gap among the four parts of the line as finally printed. The default gap is 3.

`–o` *only*
Uses as keywords any words given in the *only* file.

`–i` *ignore*
Do not use as keywords any words given the *ignore* file. If the `–i` and `–o` options are missing, use `/usr/lib/dwb/eign` as the *ignore* file.

`–b` *break*
Uses the characters in the *break* file to separate words. Tab, newline, and space characters are *always* used as break characters.

`–r` Takes any leading non-blank characters of each input line to be a reference identifier (as to a page or chapter), separate from the text of the line. Attaches that identifier as a 5th field on each output line.

NOTES

Line length counts do not account for overstriking or proportional spacing. Lines that contain tildes (~) are botched because `ptx` uses that character internally. `ptx` does not discard non-alphanumeric characters.

FILES

`/bin/sort`
`/usr/lib/dwb/eign`

```
/usr/lib/tmac/tmac.ptx
```

SEE ALSO

nroff(1M), troff(1M), mptx(5).

System Administrator's Reference Manual

putdev(1M)

NAME

`putdev -` edit device table

SYNOPSIS

`putdev –a` *alias* [*attribute=value* [. . .]]

`putdev –m` *device attribute=value* [*attribute=value*[. . .]]

`putdev –d` *device* [*attribute* [. . .]]

DESCRIPTION

`putdev` can add a new device to the device table, modify an existing device description or remove a device entry from the table. The first synopsis is used to add a device. The second synopsis is used to modify existing entries by adding or changing attributes. If a specified attribute is not defined, this option adds that attribute to the device definition. If it is already defined, it modifies the attribute definition. The third synopsis is used to delete either an entire device entry or, if the attribute argument is used, to delete an attribute assignment for a device.

OPTIONS

`-a` Adds a device to the device table using the specified attributes. The device must be referenced by its *alias*.

`-m` Modifies a device entry in the device table. If an entry already exists, it adds any specified attributes that are not defined. It also modifies any attributes which already have a value with the value specified with this command.

`-d` Removes a device from the device table, when executed without the *attributes* argument. Used with the *attribute* argument, it deletes the given attribute specification for *device* from the table.

alias
Designates the alias of the device to be added.

device
Designates the pathname or alias of the device whose attribute is to be added, modified, or removed.

attribute
Designates a device attribute to be added or modified. Can be any of the device attributes described under **NOTES** except alias. This prevents an accidental modification or deletion of a device's alias from the table.

value
Designates the value to be assigned to a device's attribute.

NOTES

The following list shows all of the attributes which can be defined for a device:

`alias`
The unique name by which a device is known. No two devices in the database may share the same alias name. The name is limited in length to 14 characters and should contain only alphanumeric characters and also the following special characters if they are escaped with a backslash: underscore (_), dollar sign ($), hyphen (-), and period (.).

`bdevice`
The pathname to the block special device node associated with the device, if any. The associated major/minor combination should be unique within the database and should match that associated with the `cdevice` field, if any. (It is the administrator's responsibility to ensure that these `major/minor` numbers are unique in the database.)

`capacity`

The capacity of the device or of the typical volume, if removable.

`cdevice`
The pathname to the character special device node associated with the device, if any. The associated major/minor combination should be unique within the database and should match that associated with the `bdevice` field, if any. (It is the administrator's responsibility to ensure that these `major/minor` numbers are unique in the database.)

`cyl` Used by the command specified in the `mkfscmd` attribute.

`desc` A description of any instance of a volume associated with this device.

`dpartlist`
The list of disk partitions associated with this device. Used only if `type=disk`. The list should contain device aliases, each of which must have `type=dpart`.

`dparttype`
The type of disk partition represented by this device. Used only if `type=dpart`. It should be either `fs` (for filesystem) or `dp` (for data partition).

`erasecmd`
The command string that, when executed, erases the device.

`fmtcmd`
The command string that, when executed, formats the device.

`fsname`
The filesystem name on the file system administered on this partition, as supplied to the `/usr/sbin/labelit` command. This attribute is specified only if `type=dpart` and `dparttype=fs`.

`gap` Used by the command specified in the `mkfscmd` attribute.

`mkfscmd`
The command string that, when executed, places a file system on a previously formatted device.

`mountpt`
The default mount point to use for the device. Used only if the device is mountable. For disk partitions where `type=dpart` and `dparttype=fs`, this attribute should specify the location where the partition is normally mounted.

`nblocks`
The number of blocks in the filesystem administered on this partition. Used only if `type=dpart` and `dparttype=fs`.

`ninodes`
The number of inodes in the filesystem administered on this partition. Used only if `type=dpart` and `dparttype=fs`.

`norewind`
The name of the character special device node that allows access to the serial device without rewinding when the device is closed.

`pathname`
Defines the pathname to an i-node describing the device (used for non-block or character device pathnames, such as directories).

`type` A token that represents inherent qualities of the device. Standard types include: 9-track, disk, directory, dpart, and qtape.

`volname`
The volume name on the filesystem administered on this partition, as supplied to the `/usr/sbin/labelit` command. Used only if `type=dpart` and `dparttype=fs`.

`volume`
A text string used to describe any instance of a volume associated with this device. This attribute should not be defined for devices which are not removable.

DIAGNOSTICS

The command will exit with one of the following values:

`0` = Successful completion of the task.

`1` = Command syntax incorrect, invalid option used, or internal error occurred.

`2` = Device table could not be opened for reading or new device table could not be created.

`3` = If executed with the `-a` option, indicates that an entry in the device table with the alias *alias* already exits. If executed with the `-m` or `-d` options, indicates that no entry exists for device *device*.

`4` = Indicates that `-d` was requested and one or more of the specified attributes were not defined for the device.

FILES

`/etc/device.tab`

SEE ALSO

`devattr`(1M), `putdgrp`(1M).

System Administrator's Reference Manual

putdgrp(1M)

NAME

putdgrp - edit device group table

SYNOPSIS

putdgrp [–d] *dgroup* [*device* [. . .]]

DESCRIPTION

putdgrp modifies the device group table. It performs two kinds of modification. It can modify the table by creating a new device group or removing a device group. It can also change group definitions by adding or removing a device from the group definition.

When the command is invoked with only a *dgroup* specification, the command adds the specified group name to the device group table if it does not already exist. If the –d option is also used with only the *dgroup* specification, the command deletes the group from the table.

When the command is invoked with both a *dgroup* and a *device* specification, it adds the given device name (or names) to the group definition. When invoked with both arguments and the –d option, the command deletes the device name (or names) from the group definition.

When the command is invoked with both a *dgroup* and a *device* specification and the device group does not exist, it creates the group and adds the specified devices to that new group.

OPTIONS

–d Deletes the group or, if used with *device*, the device from a group definition.

dgroup
Specifies a device group name.

device
Specifies the pathname or alias of the device that is to added to or deleted from the device group.

DIAGNOSTICS

The command will exit with one of the following values:

0 = Successful completion of the task.

1 = Command syntax incorrect, invalid option used, or internal error occurred.

2 = Device group table could not be opened for reading or a new device group table could not be created.

3 = If executed with the –d option, indicates that an entry in the device group table for the device group *dgroup* does not exist and so cannot be deleted. Otherwise, indicates that the device group *dgroup* already exists and cannot be added.

4 = If executed with the –d option, indicates that the device group *dgroup* does not have as members one or more of the specified devices. Otherwise, indicates that the device group *dgroup* already has one or more of the specified devices as members.

EXAMPLE

To add a new device group:

```
putdgrp sdisk □
```

To add a device to a device group:

```
putdgrp sdisk /dev/ios0/sdisk003s0 □
```

To delete a device group:

```
putdgrp -d sdisk □
```

To delete a device from a device group:

```
putdgrp -d sdisk /dev/ios0/sdisk003s0
```

FILES

`/etc/dgroup.tab`

SEE ALSO

`listdgrp`(1M), `putdev`(1M).

System Administrator's Reference Manual

pwck(1M-ucb)
BSD Compatibility Package

NAME

`pwck` - check password database entries

SYNOPSIS

`/usr/ucb/pwck` [*filename*]

DESCRIPTION

`pwck` checks a password file for errors. If specified, *filename* is checked, otherwise `/etc/passwd` is checked.

This command differs from `/usr/sbin/pwck` in its ability to correctly parse YP entries in `/etc/passwd`.

DIAGNOSTICS

`Too many/few fields`
: An entry in the password file does not have the proper number of fields.

`No login name`
: The login name field of an entry is empty.

`Bad character(s) in login name`
: The login name in an entry contains characters other than lowercase letters and digits.

`First char in login name not lowercase alpha`
: The login name in an entry does not begin with a lowercase letter.

`Login name too long`
: The login name in an entry has more than 8 characters.

`Invalid UID`
: The user ID field in an entry is not numeric or is greater than 65535.

`Invalid GID`
: The group ID field in an entry is not numeric or is greater than 65535.

`No login directory`
: The login directory field in an entry is empty.

`Login directory not found`
: The login directory field in an entry refers to a directory that does not exist.

`Optional shell file not found.`
: The login shell field in an entry refers to a program or shell script that does not exist.

`No netgroup name`
: The entry is a YP entry referring to a netgroup, but no netgroup is present.

`Bad character(s) in netgroup name`
: The netgroup name in a YP entry contains characters other than lowercase letters and digits.

`First char in netgroup name not lowercase alpha`
: The netgroup name in a Yellow pages entry does not begin with a lowercase letter.

FILES

`/etc/passwd`

SEE ALSO

`group(4)`, `passwd(4)`.

System Administrator's Reference Manual

pwck(1M)

NAME

pwck, grpck - password/group file checkers

SYNOPSIS

/usr/sbin/pwck [*file*]

/usr/sbin/grpck [*file*]

DESCRIPTION

pwck scans the password file and notes any inconsistencies. The checks include validation of the number of fields, login name, user ID, group ID, and whether the login directory and the program-to-use-as-shell exist. The default password file is /etc/passwd.

grpck verifies all entries in the group file. This verification includes a check of the number of fields, group name, group ID, whether any login names belong to more than NGROUPS_MAX groups and that all login names appear in the password file. The default group file is /etc/group.

DIAGNOSTICS

Group entries in /etc/group with no login names are flagged.

LOCALE

The LC_MESSAGES environment variable governs the language in which message texts are displayed.

The LC_CTYPE environment variable governs character classes, character conversion (shifting) and the behavior of character classes in regular expressions.

If LC_MESSAGES or LC_CTYPE is undefined or is defined as the null string, it defaults to the value of LANG. If LANG is likewise undefined or null, the system acts as if it were not internationalized.

If any of the locale variables has an invalid value, the system acts as if none of the variables was set.

The LC_ALL environment variable governs the entire locale. LC_ALL takes precedence over all the other environment variables that affect internationalization.

FILES

/etc/group

/etc/passwd

SEE ALSO

group(4), passwd(4).

System Administrator's Reference Manual

pwconv(1M)

NAME

`pwconv` - install and update /etc/shadowwith information from /etc/passwd

SYNOPSIS

`pwconv`

DESCRIPTION

The `pwconv` command creates and updates `/etc/shadow` with information from `/etc/passwd`.

If the `/etc/shadow` file does not exist, this command will create `/etc/shadow` with information from `/etc/passwd`. The command populates `/etc/shadow` with the user's login name, password, and password aging information. If password aging information does not exist in `/etc/passwd` for a given user, none will be added to `/etc/shadow`. However, the last changed information will always be updated.

If the `/etc/shadow` file does exist, the following tasks will be performed:

- Entries that are in the `/etc/passwd` file and not in the `/etc/shadow` file will be added to the `/etc/shadow` file.
- Entries that are in the `/etc/shadow` file and not in the `/etc/passwd` file will be removed from `/etc/shadow`.
- Password attributes (e.g. password and aging information) that exist in an `/etc/passwd` entry will be moved to the corresponding entry in `/etc/shadow`.

The `pwconv` program is a privileged system command that cannot be executed by ordinary users.

DIAGNOSTICS

The `pwconv` command exits with one of the following values:

`0` = Success.

`1` = Permission denied.

`2` = Invalid command syntax.

`3` = Unexpected failure. Conversion not done.

`4` = Unexpected failure. Password file(s) missing.

`5` = Password file(s) busy. Try again later.

FILES

```
/etc/passwd
/etc/shadow
/etc/opasswd
/etc/oshadow
```

SEE ALSO

`passwd(1)`, `passmgmt(1M)`.

quickdd(1M) - runacct(1M)

System Administrator's Reference Manual

quickdd(1M)

NAME

`quickdd` - copy a file

SYNOPSIS

`quickdd` [*options ...*]

DESCRIPTION

`quickdd` is a fast version of `dd(1)` which forks a read and a write process to achieve a high-performance data transfer. This utility uses the same syntax as `dd`, but does not support any data conversions. The transferred date for the two processes is held in a large circular buffer in shared memory. Process synchronization is performed with IPC (InterProcess Communication) semaphores.

Standard input and output are used by default. The input and output block sizes can be specified to take advantage of raw physical I/O.

OPTIONS

`if=`*file*
: input file name; standard input is the default.

`of=`*file*
: output file name; standard output is the default.

`ibs=`*n*
: input block size *n* bytes (default 32768).

`obs=`*n*
: output block size *n* bytes (default 32768).

`bs=`*n* sets both input and output block size, superseding `ibs` and `obs`.

`skip=`*n*
: skips *n* input blocks before starting copy (appropriate for magnetic tape, where `iseek` is not defined).

`iseek=`*n*
: seeks *n* blocks from beginning of input file before copying (appropriate for disk files, where `skip` can be incredibly slow).

`oseek=`*n*
: seeks *n* blocks from beginning of output file before copying.

`seek=`*n*
: identical to `oseek`, retained for backward compatibility.

`count=`*n*
: copies only *n* input blocks.

`conv=sync`
: pads every input block to `ibs`, if `ibs` is equal to `obs`. No other conversions are supported.

Where sizes are specified, a number of bytes is expected. A number can end with `k`, `b`, or `w` to specify multiplication by 1024, 512, or 2 respectively; a pair of numbers may be separated by `x` to indicate multiplication.

After completion, `quickdd` reports the number of whole and partial blocks, the number of transferred bytes, and the transfer rate in kilobytes/s for input and output. The transfer rates are measured irrespective of the read and write processes and include the elapsed time for all reads or writes.

NOTES

Do not use `quickdd` to copy files between file systems with different block sizes. The best performance depends on the type of the raw device and will be achieved with block sizes between 8 and 128 KBytes.

Using a blocked device to copy a file results in extra nulls being added to the file to pad the final block to the block boundary.

`ibs` and `obs` may differ, but `quickdd` will object if one of them is not aligned to 4.

Because IPC resources are being used (shared memory and semaphores), these resources will not be removed if `quickdd` is killed with the signal `SIGKILL`. In this case, remove these resources with `ipcrm`.

DIAGNOSTICS

f+*p* records in(out) *t* bytes *tr* kB/s □
numbers of *f*ull and *p*artial blocks read(written), *t*otal bytes and the *t*ransfer *r*ate.

Example:

```
1000+0 records  in,      8192000 bytes  171.67 kB/s
1000+0 records out,      8192000 bytes  233.24 kB/s
```

SEE ALSO

cp(1), dd(1), ipcrm(1).

System Administrator's Reference Manual

quot(1M)

NAME

`quot` - summarize file system ownership

SYNOPSIS

`quot` [`–achnv`] [*filesystem*]

`quot` [`–afhn`] [*filesystem*]

DESCRIPTION

`quot` displays the number of blocks (1024 bytes) in the named *filesystem* currently owned by each user. There is a limit of 2048 blocks. Files larger than this will be counted as a 2048 block file, but the total blocks count will be correct.

OPTIONS

`–a` Generates a report for all mounted file systems. The *filesystem* specification has no effect here.

`–c` Displays three columns giving a file size in blocks, the number of files of that size, and a cumulative total of blocks containing files of that size or a smaller size.

`–f` Displays for all users how many files belong to them, and how much space in Kbytes they are occupying. This options is incompatible with the `–c` and `–v` options.

`–h` Estimates the number of blocks in the file – this does not account for files with holes in them.

`–n` Attaches names to the list of files read from standard input. `quot –n` cannot be used alone, because it expects data from standard input. For example, the pipeline "`ncheck` *filesystem* | `sort +0n` | `quot –n` *filesystem*" will produce a list of all files and their owners. This option is incompatible with all other options.

`–v` In addition to the default output, displays three columns containing the number of blocks not accessed in the last 30, 60, and 90 days.

NOTES

This command may only be used by a privileged user.

FILES

`/etc/mnttab`
mounted file systems

`/etc/passwd`
to get user names

SEE ALSO

`du`(1) .

System Administrator's Reference Manual

quota(1M)

NAME

`quota` - display a user's disk quota and usage

SYNOPSIS

`quota` [`–v`] [*username*]

DESCRIPTION

`quota` displays users' disk usage and limits. Only a privileged user may use the optional *username* argument to view the limits of other users.

`quota` without options displays only warnings about mounted file systems where usage is over quota. Remotely mounted file systems which do not have quotas turned on are ignored.

username can be numeric, corresponding to the uid of a user.

The `–v` option displays user's quotas on all mounted file systems where quotas exist.

FILES

`/etc/mnttab`
: list of currently mounted filesystems

SEE ALSO

`edquota`(1M), `quotaon`(1M).

System Administrator's Reference Manual

quotacheck(1M)

NAME

`quotacheck` - file system quota consistency checker

SYNOPSIS

`quotacheck` [–v] [–p] *filesystem* ...

`quotacheck` [–apv]

DESCRIPTION

`quotacheck` examines each file system, builds a table of current disk usage, and compares this table against that stored in the disk quota file for the file system. If any inconsistencies are detected, both the quota file and the current system copy of the incorrect quotas are updated (the latter only occurs if an active file system is checked).

`quotacheck` expects each file system to be checked to have a quota file named `quotas` in the root directory. If none is present, `quotacheck` will ignore the file system.

`quotacheck` accesses the character special device in calculating the actual disk usage for each user. Thus, the file systems checked should be quiescent while `quotacheck` is running.

OPTIONS

–v Indicates the calculated disk quotas for each user on a particular file system. `quotacheck` normally reports only those quotas modified.

–a Checks the file systems indicated in `/etc/mnttab` to be read-write with disk quotas. Only those file systems that have "rq" in the `mntopts` field of the `/etc/vfstab` file are checked.

–p Runs parallel passes on the required file systems.

FILES

`/etc/mnttab`
mounted file systems

`/etc/vfstab`
list of default parameters for each file system

SEE ALSO

`quotaon`(1M) .

System Administrator's Reference Manual

quotaon(1M)

NAME

quotaon, quotaoff - turn file system quotas on and off

SYNOPSIS

quotaon [–v] *filesystem* ...

quotaon [–av]

quotaoff [–v] *filesystem* ...

quotaoff [–av]

DESCRIPTION

quotaon announces to the system that disk quotas should be enabled on one or more file systems. The file systems specified must be mounted at the time. The file system quota files must be present in the root directory of the specified file system and be named quotas.

quotaoff announces to the system that file systems specified should have any disk quotas turned off.

OPTIONS

–a All file systems in /etc/mnttab marked read-write with quotas will have their quotas turned on. This option is normally used at boot time to enable quotas. It applies only to file systems that have "rq" in the mntopts field of the /etc/vfstab file.

–v Displays a message for each file system where quotas are turned on.

The options for quotaoff are:

–a Forces all file systems in /etc/mnttab to have their quotas disabled. This option applies only to file systems that have "rq" in the mntopts field of the /etc/vfstab file.

–v Displays a message for each file system affected.

These commands update the status field of devices located in /etc/mnttab to indicate when quotas are on or off for each file system.

FILES

/etc/mnttab
mounted file systems

/etc/vfstab
list of default parameters for each file system

SEE ALSO

mnttab(4), vfstab(4).

System Administrator's Reference Manual

rc0(1M)

NAME

`rc0` - run commands performed to stop the operating system

SYNOPSIS

`/sbin/rc0`

DESCRIPTION

This file is executed at each system state change that needs to have the system in an inactive state. It is responsible for those actions that bring the system to a quiescent state, traditionally called "shutdown".

There are two system states that require this procedure. They are state `0` (power off), state `5` (the SASH state), and state `6` (the reboot state). Whenever a change to one of these states occurs, the `rc0` procedure is run. The entry in `/etc/inittab` might read:

```
s0:056:wait:/sbin/rc0 >/dev/console 2>&1 </dev/console
```

Some of the actions performed by `rc0` are carried out by files in the directory `/etc/shutdown.d`. and files beginning with `K` in `/etc/rc0.d`. These files are executed in ASCII order (see **FILES** below for more information), terminating some system service. The combination of commands in `rc0` and files in `/etc/shutdown.d` and `/etc/rc0.d` determines how the system is shut down.

The recommended sequence for `rc0` is:

- Stop System Services and Daemons. Various system services (such as a Local Area Network or LP Spooler) are gracefully terminated.

  When new services are added that should be terminated when the system is shut down, the appropriate files are installed in `/etc/shutdown.d` and `/etc/rc0.d`.
- Terminate Processes `SIGTERM` signals are sent to all running processes by `killall(1M)`. Processes stop themselves cleanly if sent `SIGTERM`.
- Kill Processes `SIGKILL` signals are sent to all remaining processes; no process can resist `SIGKILL`.

  At this point the only processes left are those associated with `rc0` and processes `0` and `1`, which are special to the operating system.
- Unmount All File Systems Only `/root`, `proc`, `/stand`, and `/dev/fdfs` remain mounted.

Depending on which system state the systems end up in (`0`, `5`, or `6`), the entries in `/etc/inittab` will direct what happens next. If the `/etc/inittab` has not defined any other actions to be performed as in the case of system state `0`, then the operating system will have nothing to do. It should not be possible to get the system's attention. The command can be used only by the superuser.

FILES

The execution by `/usr/bin/sh` of any files in `/etc/shutdown.d` occurs in ASCII sort-sequence order. See `rc2(1M)` for more information.

SEE ALSO

`killall(1M)`, `rc2(1M)`, `shutdown(1M)`.

System Administrator's Reference Manual

rc1(1M)

NAME

`rc1 -` run commands performed to bring the system down foradministration

SYNOPSIS

`/sbin/rc1`

DESCRIPTION

This file is executed at each system state change that needs to have the system in an administrative state. It is responsible for those actions that bring the system to an administrative state. When the system is changing to init state 1 it requires this procedure. The entry in `/etc/inittab` might read:

```
s1:1:wait:/sbin/rc1 >/dev/console 2>&1 </dev/console
```

Some of the actions performed by `rc1` are carried out by files beginning with `K` and `S` in `/etc/rc1.d`. These files are executed in ASCII order (see **FILES** below for more information), terminating some system service. The combination of commands in `rc1` and files in `/etc/rc1.d` determines how the system is shut down.

The recommended sequence for `rc1` is:

- Stop System Services and Daemons. Various system services (such as a Local Area Network or LP Spooler) are terminated gracefully.

  When new services are added that should be terminated when the system is brought down to administrative state, the appropriate files are installed in `/etc/rc1.d`.

The command can be used only by the superuser.

FILES

The execution by `/usr/bin/sh` of any files in `/etc/rc1.d` occurs in ASCII sort-sequence order. See `rc2`(1M) for more information.

SEE ALSO

`rc2`(1M) .

System Administrator's Reference Manual

rc2(1M)

NAME

`rc2` - run commands performed for multiuser environment

SYNOPSIS

`/sbin/rc2`

DESCRIPTION

This file is executed via an entry in `/etc/inittab` and is responsible for those initializations that bring the system to a ready-to-use state, traditionally state `2`, called the "multiuser" state.

The actions performed by `rc2` are found in files in the directory `/etc/rc2.d` and files beginning with `S` in `/etc/rc2.d`. These files are executed by `/usr/bin/sh` in ASCII sort-sequence order (see **FILES** for more information). When functions are added that need to be initialized when the system goes multiuser, an appropriate file should be added in `/etc/rc2.d`.

The functions done by the `rc2` command and associated `/etc/rc2.d` files include:

- Setting and exporting the `TZ` variable.
- Setting-up and mounting the user (`/usr`) file system.
- Cleaning up (remaking) the `/tmp` and `/var/tmp` directories.
- Loading the network interface with program data and starting the associated processes.
- Starting the `cron` daemon by executing `/usr/sbin/cron`.
- Cleaning up (deleting) uucp locks status, and temporary files in the `/var/spool/uucp` directory.

Other functions can be added, as required, to support the addition of hardware and software features.

EXAMPLES

The following are prototypical files found in `/etc/rc2.d`. These files are prefixed by an `S` and a number indicating the execution order of the files.

```
MOUNTFILSYS
     # Set up and mount file systems
     cd /
     /sbin/mountall -l
RMTMPFILES
     # clean up /tmp
     rm –rf /tmp
     mkdir /tmp
     chmod 777 /tmp
     chgrp sys /tmp
     chown sys /tmp
uucp  # clean-up uucp locks, status, and temporary files
     rm –rf /var/spool/locks/*
```

The file `/etc/TZ` is included early in `rc2`, thus establishing the default time zone for all commands that follow.

NOTES

All actions in files beginning with `S` in `/etc/rc2.d` can also be stopped and restarted manually using `sh`. To do this, call:

`sh /etc/rc2.d/`*`file`* `stop`

or

`sh /etc/rc2.d/`*file* `start`

FILES

Here are some hints about files in `/etc/rc2.d`:

The order in which files are executed is important. Since they are executed in ASCII sort-sequence order, using the first character of the file name as a sequence indicator will help keep the proper order. Thus, files starting with the following characters would be:

[`0-9`]
: Very early

[`A-Z`]
: Early

[`a-n`]
: Later

[`o-z`]
: Last

Files in `/etc/rc2.d` that begin with a dot (`.`) will not be executed. This feature can be used to hide files that are not to be executed for the time being without removing them. The command can be used only by the superuser.

Files in `/etc/rc2.d` must begin with an `S` or a `K` followed by a number and the rest of the file name. Upon entering run level 2, files beginning with `S` are executed with the `start` option; files beginning with `K`, are executed with the `stop` option. Files beginning with other characters are ignored.

SEE ALSO

`shutdown`(1M) .

System Administrator's Reference Manual

rc6(1M)

NAME

`rc6 -` run commands performed to stop and reboot the operating system

SYNOPSIS

`/sbin/rc6`

DESCRIPTION

The shell script `rc6` is run whenever a transition to run state 6 is requested either through `init 6` or `shutdown –i6`.

The sequence of events in `rc6` is as follows:

- check to see if there is a new operating system to be rebooted, via the existence of the file `/etc/.new_unix`. If this file exists, it carefully moves the kernel from `/etc/conf/cf.d/unix` into `/stand/unix` and links `/unix` to it. Any errors will result in an attempt to restore things.
- unmount all file systems, then reboot the system. The system will come up in a state defined by the `initdefault` entry in `/etc/inittab`.

SEE ALSO

`init`(1M), `rc0`(1M), `rc2`(1M), `shutdown`(1M), `inittab`(4).

System Administrator's Guide.

System Administrator's Reference Manual

rcS(1M)

NAME

`rcS` - run commands performed to go to singleuser mode

SYNOPSIS

`/sbin/rcS`

DESCRIPTION

This file is executed via an entry in `/etc/inittab` and is responsible for those initializations that bring the system to a singleuser state, traditionally state `S`. Refer to `init`(1M) man page for complete description of singleuser state. Functions not handled by `init`(1M) when it is going to singleuser state are performed by `/sbin/rcS`. For example, a process not started by `init`, like `cron`, will keep running. This can be used to kill processes previously established by each site.

The actions performed by `rcS` are found in files beginning with `K` in `/etc/rcS.d`. These files are executed by `/usr/bin/sh` in ASCII sort-sequence order (see **FILES** for more information). When functions are added that need to be initialized when the system goes multiuser, an appropriate file should be added in `/etc/rcS.d`.

Other functions can be added, as required, to support the addition of software features.

EXAMPLES

The following are prototypical files found in `/etc/rcS.d`. These files are prefixed by an `K` and a number indicating the execution order of the files.

K00ANNOUNCE	K05fmtd	K20motif	K40nfs	K65snmpd	K74smtpd
K02nerv	K11osm	K22acct	K41rwho	K69inet	K75syslog
K03news	K15termadm	K39yp	K50rpc	K70cron	

FILES

Files in `/etc/rcS.d` must begin with a `K` followed by a number and the rest of the file name. When the system is coming down to run level `S`, files beginning with `K`, are executed with the `stop` option. Files beginning with other characters are ignored.

SEE ALSO

`shutdown`(1M) .

System Administrator's Reference Manual

reboot(1M)
BSD Compatibility Package

NAME

`reboot` - restart the operating system

SYNOPSIS

`/usr/ucb/reboot` [`–dnq`] [*boot_arguments*]

DESCRIPTION

`reboot` restarts the kernel. The kernel is loaded into memory by the PROM monitor, which transfers control to it.

Although `reboot` can be run by the privileged user at any time, `shutdown` is normally used first to warn all users logged in of the impending loss of service. See `shutdown(1M)` for details.

`reboot` performs a `sync(1M)` operation on the disks, and then a multiuser reboot is initiated. See `init(1M)` for details.

`reboot` normally logs the reboot to the system log daemon, `syslogd(1M)` and places a shutdown record in the login accounting file `/var/adm/wtmp`. These actions are inhibited if the `–n` or `–q` options are present.

OPTIONS

`–d` Dumps system core before rebooting. This option is provided for compatibility, but is not supported by the underlying `reboot` call.

`–n` Avoids the `sync(1M)`. It can be used if a disk or the processor is on fire.

`–q` Quick. Reboots quickly and ungracefully, without first shutting down running processes.

boot_arguments
These arguments are accepted for compatibility, but are ignored by `reboot`.

Power Fail and Crash Recovery

Normally, the system will reboot itself at power-up or after crashes.

FILES

`/var/adm/wtmp`
Login accounting file

SEE ALSO

`crash(1M)`, `fsck(1M)`, `halt(1M)`, `init(1M)`, `shutdown(1M)`, `sync(1M)`, `syslogd(1M)`, `sync(2)`, `reboot(3)`.

System Administrator's Reference Manual

removef(1M)

NAME

`removef` - remove a file from software database

SYNOPSIS

`/usr/sbin/removef` *pkginst path1* [*path2* ...]

`/usr/sbin/removef –f` *pkginst*

DESCRIPTION

`removef` informs the system that the user, or software, intends to remove a pathname. Output from `removef` is the list of input pathnames that may be safely removed (no other packages have a dependency on them).

After all files have been processed, `removef` should be invoked with the `–f` option to indicate that the removal phase is complete.

OPTIONS

`–f` Indicates that the removal phase is complete.

pkginst
Defines the package instance with which the pathnames(s) is (are) to be associated.

path The pathname(s) to be removed.

EXAMPLE

The following shows the use of `removef` in an optional pre-install script:

```
echo "The following files are no longer part of this package
 and are being removed."
removef $PKGINST /dev/xt[0-9][0-9][0-9] |
while read pathname
do
 echo "$pathname"
 rm –f $pathname
done
removef –f $PKGINST || exit 2
```

SEE ALSO

`installf`(1M), `pkgadd`(1M), `pkgask`(1M), `pkgchk`(1M), `pkginfo`(1M), `pkgmk`(1M), `pkgparam`(1M), `pkgproto`(1M), `pkgtrans`(1M).

System Administrator's Reference Manual

repquota(1M)

NAME

`repquota` - summarize quotas for a file system

SYNOPSIS

`repquota` [–v] *filesystem* ...

`repquota` [–av]

DESCRIPTION

`repquota` prints a summary of the disk usage and quotas for the specified file systems. For each user the current number of files and amount of space (in kilobytes) is printed, along with any quotas created with `edquota`.

OPTIONS

–a Reports all file systems that have "rq" in the `mntopts` field of the `/etc/vfstab` file.

–v Reports all quotas, even if there is no usage.

Only privileged users may view quotas which are not their own.

SEE ALSO

`edquota`(1M), `quota`(1M), `quotacheck`(1M), `quotaon`(1M).

System Administrator's Reference Manual

reset_lar(1M)
RM600 only

NAME

`reset_lar -` activate controller/delete attributes

SYNOPSIS

`reset_lar –c` *cabinet_ID* `–b` *bus_ID* [`–s` *slot_ID*] [`–n` *channel*] [`–A` *attr*]

DESCRIPTION

The `reset_lar` command deletes attributes in the `HW_CONFIG` configuration object. The attribute to be deleted can be selected using the `–A` *attr* option (1 = `DEFECT`, 2 = `ERROR`). Attribute 1 `DEFECT` is deleted if no attribute is specified.

If attribute 1 `DEFECT` is deleted, the following additional action is performed:

The `reset_lar` command checks whether a command to set the board to `OK` is specified for this board type in the `ctlr_defect.tab` table. If this is the case, this command is executed.

OPTIONS

`–A` *attr*
: Specifies the attribute number (1 or 2). Default: 1.

`–c` *cabinet_ID*
: Specifies the logical cabinet number.

`–b` *bus_id*
: Specifies the bus ID.

`–s` *slot_ID*
: Specifies the slot number of the controller.

`–n` *channel*
: Specifies the channel number on the controller.

DIAGNOSTICS

Return values have the following meaning:

`1 =` Invalid parameter or option

`2 =` An error occurred during editing of `HW_CONFIG`

NOTES

If a slot is not operational, its channels also cannot be accessed.

FILES

`/usr/bin/lar/ctlr_defect.tab`
`/etc/init.d/lar_rc`
`/var/lar/hwconfig`

SEE ALSO

`act_ctlr`(1M), `act_dev`(1M), `set_lar`(1M), `lar`(7), `olr`(7).

System Administrator's Reference Manual

rtyadm(1M)

NAME

`rtyadm -` administer the RTTY monitor

SYNOPSIS

`/usr/sbin/rtyadm –d` *device_path* `–h` *host* `–t` *tcp_port* [`–r` *repeat_cnt*] [`–c` *device_conn_mode*] [`–k`] [`–f`] [`–p`] [`–e`] [`–u` *hup_mode*] [`–i` *init_string*] [`–s` *service*] [`–m` *modules*]

`/usr/sbin/rtyadm –T` *level pmtag*

`/usr/sbin/rtyadm –S` [`–i` *svctag*] *pmtag*

`/usr/sbin/rtyadm –V`

`/usr/sbin/rtyadm –?`

DESCRIPTION

`rtyadm` makes is easier to administer RTTY monitors. `rtyadm` supports various commands:

- The first syntax outputs the RTTY monitor-specific entry for the file `_pmtab` in the correct format in `stdout`.
- The `–S` option displays the status of the connections being monitored.
- The `–T` option selects (and terminates) trace mode.
- The versions command with the `–V` option returns the version number of the `_pmtab`-specific format to `stdout`.

There is a range of options and arguments for each call, which are described in the following section.

Format output

For a more detailed description of their effect, please see `rtymon`(1M) . The names of the arguments correspond to the names of the fields in the `_pmtab` file.

`–d` *device_path*
: This is the path name of the *user channel* via which the LAN device can be addressed. The path name must be specified absolutely. A check is run to determine whether a character-mode device (TTY) is located at this path name.

`–h` *host*
: This option specifies the host to be connected to the network. A value must be specified.

`–t` *tcp_port*
: This option specifies the tcp port number. A value must be specified (the value must be an integer).

`–r` *repeat_cnt*
: The retry counter in case of errors. If this option is not specified, the field is occupied with `–1`. This means that the number of retries is unlimited.

`–c` *device_conn_mode*
: This option specifies how the connection to the LAN device is to be set up. Permitted values are `out`, `data`, and `in`. If this option is not specified, `out` is assumed.

`–k` The flag `k` for the KeepAlive option is set in the `_pmtab` field `CONN_FLAGS`.

`–f` The flag `f` for `ON_THE_FLY` is set in the `_pmtab` field `CONN_FLAGS`.

`–p` The flag `p` for `POLL` is set in the `_pmtab` field `CONN_FLAGS`.

`–e` The flag `t` for `TELNET` is set in the `_pmtab` field `CONN_FLAGS`.

`–u` *hup_mode*
: Defines the type of host monitoring. Permitted values are `no_hup`, `late_hup`, and `imm_hup`. If the option is not specified, the `_pmtab` field `HUP_MODE` remains empty, and host monitoring does not take

place.

–i *init_string*
After the connection has been set up, the specified string is sent to the LAN device.

–s *service*
This option defines whether rtymon should call a utility before the rtymux driver is linked to the network connection.

–m *modules*
The corresponding _pmtab field has not yet been used.

Trace (–T option)

This option can be used to activate and deactivate output of debug information at a later stage.

–T *level*
The argument *level* specifies the trace level. At level 0, the trace is deactivated. For information on the different levels, see rtymon(1M) , –D option.

pmtag
The name of the rtymon under SAF.

Status output (–S option)

For each *svctag* of an rtymon, the status reached is written in stdout. As a side effect, the status and the _pmtab data for each *pmtag* are written under /var/saf/*pmtag*/devices and host information, if there is any, is written under /var/saf/*pmtag*/hosts (provided host checking is activated).

–i *svctag*
If this option is used, only the status of the specified *svctag* is output. In addition to the status, _pmtab data for the *svctag* is also output.

pmtag
The name of the rtymon under SAF for which specifications must be made.

Output of the _pmtab version (–V option)

The version of the file _pmtab is output in stdout as required for pmadm -a.

–? (help) option

The command syntax is displayed in stdout.

WARNING

Trace mode should be activated for short periods only, so that the trace file does not become too large.

DIAGNOSTICS

If no errors occur, rtyadm terminates with the return value 0. If errors occur, the return values have the following meanings:

\1 = Syntax error.

2 = The specified *pmtag* is not an rtymon.

3 = Illegal value in an argument.

4 = The specified *svctag* could not be found.

5 = System call error.

FILES

/etc/saf/_sactab

/etc/saf/*pmtag*/_pmtab

/var/saf/*pmtag*/devices

/var/saf/*pmtag*/hosts

SEE ALSO

pmadm(1M), rtymon(1M), sac(1M), sacadm(1M), rtty(7).

System Administrator's Reference Manual

rtydb(1M)

NAME

`rtydb` - RTTY multiplexer online debugger

SYNOPSIS

`/opt/diag.d/bin/rtydb` [–h *hostname*] [–p *port*]

`/opt/diag.d/bin/rtydb –?`

DESCRIPTION

`rtydb` provides an online debugger for the RTTY multiplexer. This allows interactive execution of individual system calls on different channels of the multiplexer.

OPTIONS

If the `rtydb` command is called without arguments, the debugger registers using its version identifier and prompt "`rtydb=>` ". The address of the port and of the host to which a connection may be set up is not yet fixed, and can be queried interactively if necessary (using the command `change` or `connect`). The arguments are explained as follows:

–h *hostname*
: This option sets the name of the host to which the RTTY device is connected in the argument *hostname*. This name can be specified both symbolically and in Internet notation.

–p *port*
: This option specifies the port number of the RTTY device to be addressed in the argument *port*. A plausibility check is not carried out.

COMMANDS

The `rtydb` debugger recognizes the following channels by name:

`dbg` Represents the *debug channel* of an RTTY connection.

`adm` Represents the *admin channel* of an RTTY connection.

`conn` Represents the *socket* of an RTTY connection.

`usr` Represents the *user channel* of an RTTY connection.

`chan0`, `chan1`, `chan2`, `chan3`, `chan4`
: Freely selectable channels.

After its prompt "`rtydb=>` ", the `rtydb` debugger awaits a command. This command is either a system call or a debugger-specific command.

System call commands

`open` This opens a character special device node and assigns one of the channel names listed above. The syntax corresponds to C syntax. Permitted open flags are keywords separated by "|":

`O_RDWR | O_RDONLY | O_WRONLY | O_NDELAY`

`close`
: As an argument to this command, the channel name of the channel to be closed is specified.

`read` The syntax corresponds to C syntax, except that the channel name is specified instead of a file descriptor. As a buffer address, the symbolic names "`buffer1`" and "`buffer2`" are selected. The command `look` can be used to display the selected buffer.

`write`
: The syntax corresponds to C syntax, except that the channel name is specified instead of a file descriptor. As a buffer address, the symbolic names "`buffer1`" and "`buffer2`" are selected. The command `fill` can be used to occupy the selected buffer.

`ioctl`

This command can be used to cancel both standard commands and RTTY-specific commands (on the *debug channel*).

The `ioctl` commands with the two arguments *channel_name* and *command_name* are implemented:

`I_POP`
Deactivate the uppermost STREAMS module.

`TCSETA`
Load the selected channel with the internal debugger `termio` buffer.

`TCGETA`
Load the internal debugger `termio` buffer with the line data of the selected channel.

`TCSETS TCSETSW`
Load the selected channel with the internL debugger `termios` buffer.

`TCGETS`
Load the internal debugger *termios* buffer with the line data for the selected channel.

`TIO_OPEN`
Activate "on the fly" mode.

`TIO_TELNET`
Activate TELNET mode. Kernel will append a "\0" to "\r" (carriage return) or duplicate the IAC character (0xff). Incoming duplicated IAC or "\0" after "\r" will be deleted.

`DBGLINE`
Status display of the RTTY connection on the *debug channel*.

`DBGRESET`
Reset the trace level to 0 (*debug channel* only).

`DBGFREEON`
Activate the trace to request "message bodies" (*debug channel* only).

`DBGFREEOFF`
Deactivate the trace to request "message bodies" (*debug channel* only).

The remaining available `ioctl` commands still require a third argument. They are as follows:

`I_PUSH`
Deactivate a STREAMS module. Argument: "*module_name*"

`I_LINK`
Deactivate a multiplexer on a STREAM. Argument: "*channel_name*"

`I_UNLINK`
Remove a multiplexer from the STREAM. Argument: "*MUX_ID*"

Debugger commands

The `rtydb` debugger also recognizes a range of individual commands which control its own data structures and internal procedures. These are:

`connect`
switches a socket connection to the addressed host and port. If a host or port has not yet been set, the address is queried interactively.

`disconnect`
The socket connection set is connected.

`change`
The host and port address of the socket connection are reset interactively.

`fill(`*buf*`)`
The buffer addressed in the argument *buf* is filled with the relevant ASCII characters. The buffers are 4096 bytes in size and are called "`buffer1`" or "`buffer2`".

`look(`*buf*`)`
The buffer addressed in the argument *buf* is output. This is a block display comprising 16 lines with 16

hexadecimal characters per block. The buffers are 4096 bytes in size and are called "`buffer1`" or "`buffer2`".

`help` Provides an overview of the debugger commands.

`show` Displays the open channels and their symbolic assignments. If there is a MUX-ID or modules on the STREAM, these are also displayed.

`termio`
Displays the `cflag` in the internal debugger `termio` buffer.

`termios`
Displays the `cflag` in the internal debugger `termios` buffer.

`clearc`
Sets the `cflag` of the internal debugger `termio` and `termios` buffers to 0.

`settermio(`*arg*`)`
Sets the bits specified in the argument *arg* in the `cflag` of the `termio` and `termios` buffers. Permitted values are keywords separated by "|".

`B0 | B19200 | B9600 | HUPCL | CSTOPB | CREAD | CLOCAL | CS6 | CS7 | CS8`

`strconf`
Displays all STREAMS modules in the channels.

`q[uit]`
Quits `rtydb`.

DIAGNOSTICS

If no errors occur, `rtydb` terminates with the return value `0`. In case of error, the return values have the following meanings:

`1` = Illegal option used (`E_ILLOPT`)

SEE ALSO

`open(2)`, `close(2)`, `ioctl(2)`, `read(2)`, `write(2)`, `rtty(7)`, `termio(7)`.

System Administrator's Reference Manual

rtymon(1M)

NAME

`rtymon` - remote TTY monitor

SYNOPSIS

`/usr/lib/saf/rtymon` [`–D`] [`–f` *file*] [`–r` *Retry_time*] [`–h` *Host_check_time*] [`–d` *Delay_after_cl*] [`–c` *Closing_time*] [`–p` *Polltime_on_refused*]

`/usr/lib/saf/rtymon –?`

DESCRIPTION

`rtymon` is a port monitor based on TTY STREAMS, as prescribed in the System V Release 4 by SAF (Service Access Facility). The `rtymon` is designed to run generally only under the control of the SAC (Service Access Controller) [see `sac(1M)`]. It can be administered using the commands `sacadm(1M)` and `pmadm(1M)` .

RTTY-specific commands and formats can be controlled using the command `rtyadm(1M)` .

The purpose of the `rtymon` is to set up connections to LAN devices on which to link the `rtymux` driver. The `rtymux` driver then provides tty nodes for these LAN devices. The `rtymon` communicates with the `rtymux` driver by means of adm nodes. In this way, various events are reported to the `rtymon`, such as connection shutdown, "Last Close", etc. The behavior of an `rtymux` driver is similar to that of a local driver. Provided it is not linked to a LAN connection, the device is considered to be unavailable (no "carrier", an `open()` blocked).

The basic functions of `rtymon` are:

1) General monitor process control

Control of the monitor itself falls into this category. In the context of SAF, the monitor has two internal statuses: `ENABLED` and `DISABLED`. The monitor is started by the SAC in one of the two statuses. The status of the monitor at start-up is defined by the entry in the file `_sactab`. If the status is `DISABLED`, the monitor does not monitor any of the ports. If the status is `ENABLED`, all LAN hosts described in the `_pmtab` file are monitored. The monitor is terminated by a `SIGTERM(15)` signal.

2) Connection monitoring

In a file belonging to a monitor incarnation, `_pmtab` is described with each entry for a connection setup from a device node (`/dev/...`) to a LAN device. Each entry is provided with a unique *service identifier* (*tag*). The monitor sets up the connection to the LAN device in accordance with the entry and monitors its status. There are two types of connection setup:

a) The connection is initiated by the host (`OUT_DEVICE`).

b) The host accepts incoming connection requests (`IN_DEVICE`).

If the device is found to be activated (`DEVICE_ON`, see below, `DEVICE_CONN_MODE`), the `rtymux` driver is activated (`I_LINK`) [see `termio(7)`]. Once the LAN connection is shut down, the multiplexer is disconnected (`I_UNLINK`) [see `termio(7)`].

3) Executing tasks

In each `_pmtab` entry, a utility can be specified (e.g. loading a terminal keyboard table). The task is then carried out before linking the `rtymux` driver.

_pmtab format

In the database of the `_pmtab` monitor, there is an entry for each connection with a unique *service identifier*. Among other things, this contains an `rtymon`-specific part with the following format:

`DEVICE_PATH : HOST : TCP_PORT : REPEAT_CNT : DEVICE_CONN_MODE : CONN_FLAGS : HUP_MODE : INIT_STRING : SERVICE : MODULES : RES :`

These have the following meanings:

`DEVICE_PATH`

This field describes the name of the *user channel* of the `rtymux` driver. The LAN device for the user can be reached using the TTY node. The name must be specified absolutely.

`HOST` The host name of the LAN connection (e.g. of the terminal server or a DOS PC) must be entered in this field. A symbolic name or Internet notation (dot) can be used. A host must be specified.

`TCP_PORT`
In the case of `OUT_DEVICE`: tcp port number under which the device on the (remote) host can be reached. For TACLAN terminal servers, the values are 7001 – 7064; for DOS PCs, they are usually 7000 – 7999. In the case of `IN_DEVICE`: local tcp port number, on which incoming connection requests can be accepted. A tcp port number must be specified.

`REPEAT_CNT`
This field specifies a retry count, which indicates how often the connection setup and linking procedure is to be repeated in case of error. A pause of *retry_time* (can be set using the option `–r`, default = 90 secs.) is left between the individual repeats. 0 represents no repeat, i.e. once an error has occurred, no further attempt is made to set up the connection. –1 represents an unlimited number of repeats, and this is assumed if the field is empty.

`DEVICE_CONN_MODE`
The field specifies how the connection to the LAN device is to be set up, and when the status `DEVICE_ON` is specified. There are three different connection types:

`OUT` `OUT_DEVICE`. If the outgoing connection request was accepted, the device is considered to be activated (`DEVICE_ON` status). The following actions (service, link) are executed.

`DATA` Similar to `OUT`, except that (at least) one other character is expected from the LAN device, before the `DEVICE_ON` status is reached and service and link are executed.

`IN` `IN_DEVICE`. If an incoming connection request could be accepted, the device is considered to be activated. The `rtymon` accepts connection requests on the tcp port set in the `TCP_PORT` field. Only connection requests from the host specified in the `HOST` field are accepted. A value mut be specified.

`CONN_FLAGS`
Various flags, with which further details concerning the connection can be set.

`k` The KeepAlive option is set on the network connection to allow a periodic check of the connection, if there is no other data traffic.

`f` `ON_THE_FLY`. `rtymon` usually begins to set up all the necessary network connections as soon as it starts. This flag allows `rtymon` to set up this connection first if there is an `open()` on the *user channel*. The network connection is closed after the last `close()` from the *user channel*.

`p` `POLL`. A "Connection refused" from the remote station is not considered to be an error (e.g. a deactivated device on the TACLAN terminal server with `need_dsr = y`). After a pause of *Polltime_on_refused* (which can be set using the `–p` option; default setting = 6 secs.), the attempt is repeated. The number of retries is unlimited.

`t` `TELNET`. On establishing a new connection `rtymon` will run the telnet protocol. In this case an `IN_DEVICE` will accept an incoming connection request from an `telnet(1)` command and will serve for some protocol sequences. A daemon [see `telnetd(1M)`] is not necessary any more. After such a connection is established data will be transferred in accordance with the telnet protocol by the `rtymux` driver.

`HUP_MODE`
This field controls the type of host monitoring. In the case of `OUT_DEVICE`, host monitoring does not function unless the host allows only one connection on the specified tcp port. The principle of host monitoring for `OUT_DEVICE` is that an attempt is made to set up another connection parallel to the existing one. This should be refused ("Connection refused") as a sign that the original connection is still in place. The delay between checks is *Host_check_time* (can be set using the `–h` option; default setting = 300 secs.). In the case of `IN_DEVICE`, other connection requests are accepted for host monitoring on the local tcp port, once the first connection was accepted (and the `rtymux` driver was linked).

Host_check_time is not relevant here.

NO_HUP
: No host check. This is the default setting if the field is empty. A "silent" host logoff (due to a crash, a disturbance on the transfer line, etc.) is not noticed.

LATE_HUP
: If a new connection was established, the rtymux driver from the old (dead) connection is linked (from the point of view of the *user channel*, this corresponds to deactivation of the device, and a HANGUP is generated if stty clocal was not set). The rtymux driver is linked to the new connection.

IMM_HUP
: As above, except that the rtymux driver is already linked by the connection if the host cannot be reached ("Host down", "Host not reachable" ...). In the case of IN_DEVICE, there is no difference between this and LATE_HUP.

  This type of connection is selected if a crash or an unreachable device is considered to be the same as deactivation of a device.

INIT_STRING
: Once the connection exists, this string is sent to the LAN device. Non-printing symbols should be specified octally, e.g. \033 for ESCAPE.

SERVICE
: This field can be used to enter shell commands that are to be executed after the DEVICE_ON status and before linking. The commands are executed as a child process of the monitor by means of /sbin/sh –c "..." and can access the switched connection (STREAMS: tcp --> tirdwr --> ptem --> ldterm) using stdin, stdout, and stderr (file descriptors 0, 1, 2).

  This field could be used to load keyboard tables, for example [see keyload(1)].

MODULES
: Not realized; ignored.

RES Reserved for extensions.

An entry in the _pmtab file is made using the pmadm(1M) command and the a(dd) option. There is also the r(emove) option, which deletes an entry. The pmadm command can also be used to activate a *service identifier* [e(nable) option] or deactivate it [d(isable) option].

OPTIONS

–D
: This option increases the debug level. –D represents debug level = 1, –DD represents debug level = 2, etc. There are 5 different levels:

  0 No debug output (default).

  1 Status changes in the individual *tag*s, poll(), status of the host checks.

  2 Additional information on network connections (EOF, termination), creation of adm nodes, events from adm nodes, termination of CLOSING status, termination of an open() (in the case of ON_THE_FLY).

  3 In addition to poll-Revents, characters from the adm channel and from the SAC. Output of a child process for service, t_sndrel(), t_snddis().

  4 Additional entries in the poll array.

  Unless otherwise specified, debug outputs go to the log file.

–f *file*
: Debug outputs do not go to the log file, but to the file specified here.

–r *Retry_time*
: The time (in seconds) for which a *tag* is suspended before a further attempt is made after an error. The default is 90 seconds and 2 is assumed for values less than 2.

–h *Host_check_time*

The time delay in seconds between the individual checks of the network connections in the case of OUT_DEVICEs. The default is 300 seconds and 2 is assumed for values less than 2.

−d *Delay_after_cl*
After closing a network connection, this is the amount of time left before a new connection to this device is set up. The default value is 4 seconds and 0 is assumed for values less than 0.

−c *Closing_time*
If stty hupcl is set on the *user channel*, the network connection is closed after the last close(). This option can be used to set the time (in seconds) waited for the last characters to be transferred. Characters transferred after this time are lost. The default value is 15 seconds and 0 is assumed for values less than 0.

−p *Polltime_on_refused*
If "p" is specified in the file _pmtab in the CONN_FLAGS field: time in seconds between connection setup attempts when "Connection refused" is received from the remote station. The default value is 10 seconds and 2 is assumed for values less than 2.

DIAGNOSTICS

For diagnostic purposes, the monitor keeps a private logging file in the directory /var/saf/*pmtag* with the name log. This file can only be written by the monitor. SAC commands, status changes in the monitor, and error situations are logged in this log file.

To make it easier to trace errors, the monitor can be operated in debug or trace mode. Debug mode is activated using the −D option when the monitor is being started, or dynamically at a later stage using rtyadm −T.

Debug mode should not be activated unless it is required. Debug outputs can be very extensive (particularly if there are a lot of *tag*s), and there is no administration of the log/debug file.

WARNING

Trace modes should be activated for short periods only, so that the trace file does not become too big.

RETURN CODES

IF no errors occur, rtymon runs as a daemon. If errors occur, the return values have the following meanings:

1 = Illegal option used.

2 = Log or debug file could not be opened.

3 = Number of possible file descriptors could not be increased.

4 = Pipe for communication with SAC (../_sacpipe, _pmpipe) could not be opened.

5 = _pid file still locked; another rtymon is already running.

6 = _pid file could not be written (open, lock, or write refused).

FILES

/etc/saf/_sactab

/etc/saf/_sacpipe

/etc/saf/*pmtag*/_pmtab

/etc/saf/*pmtag*/_pmpipe

/var/saf/*pmtag*/log

/var/saf/*pmtag*/devices

/var/saf/*pmtag*/hosts

SEE ALSO

keyload(1), telnet(1), pmadm(1M), rtyadm(1M), sac(1M), sacadm(1M), telnetd(1M), open(2), connect(3N), listen(3N), rtty(7), termio(7).

System Administrator's Reference Manual

rtymux_trace(1M)

NAME

`rtymux_trace` - RTTY multiplexer trace

SYNOPSIS

`/opt/diag.d/bin/rtymux_trace –n` *lineno* [`–l` *level*]

`/opt/diag.d/bin/rtymux_trace –f`

`/opt/diag.d/bin/rtymux_trace –?`

DESCRIPTION

`rtymux_trace` activates the RTTY multiplexer trace, and trace output appears on `stdout`. There are 5 possible levels of trace data. A character special device with the name `dbg`*lineno* is sought in the `/dev/rtty` directory. This is the *debug channel* of the RTTY device. If there is already one available, it is used. Otherwise one is created. Once the `rtymux_trace` command terminates, any *debug channel* created is removed.

Trace data can only be activated from one RTTY device using this command. Since all trace data from different RTTY devices can also be output from the kernel to a special device (`/etc/.osm`), trace data for parallel `rtymux_trace` calls will appear mixed. Allocation to devices is indicated by the identifier

`RTYMUX LINE[<`*no*`>]: <`*Text*`>`

Another option for tracing is to use the command to follow the release of messages with the multiplexer. Only the address of the message block released and its type are displayed. This type of trace is not line-specific.

OPTIONS

`–n` *lineno*

Using this option, the line number of the RTTY device for which the trace is to be activated is specified in the argument *lineno*. The line number is identical to the minor device number of the *user channel* of the RTTY device.

Warning:

The line number must be specified (otherwise error no. 3 occurs, `E_ILLARG`).

`–l` *level*

This option is used to select the level of trace output. There are 5 levels of trace output for the RTTY multiplexer in the system kernel:

Level 0:

This level reports only the events `open`, `close`, `I_LINK`, and `I_UNLINK`. A status query on the *debug channel* is also output.

Level 1:

In addition to the trace data of level 0, handling of relevant "downstream messages" and their types are also reported.

Level 2:

In addition to the trace data of level 1, handling of relevant "upstream messages" and their types are also indicated.

Level 3:

In addition to the trace data of level 2, the contents of `M_DATA` messages are also indicated.

Level 4:

In addition to the trace data of level 3, all messages transferred by the *driver-put* and *usr-service* routine are also displayed.

Level 5:

All trace data output by the multiplexer is displayed.

The default level if the `l` option is not specified is 0.

`-f` (freeing) This option activates tracing of the release of message blocks by the multiplexer.

`-?` (help) The command syntax is displayed on `stderr`.

Once this command terminates, the trace is deactivated again.

WARNING

Trace mode should only be activated for short periods to ensure that the trace file `/etc/.osm` does not become too large.

DIAGNOSTICS

If no error occurs, `rtymux_trace` terminates with the return value `0`. In the case of an error, the return values have the following meanings:

`1` = Command can only be executed by the superuser (`E_PERM`).

`2` = Illegal option used (`E_ILLOPT`).

`3` = Command was called using an unknown argument (`E_ILLARG`).

`4` = Command cannot open the *debug channel* (`E_OPEN`).

`5` = Command cannot create the *debug channel* (`E_MKNOD`).

`6` = Command cannot start a trace process (`E_FORK`).

`7` = `stat()` on the *debug channel* failed (`E_STAT`).

`8` = There is a *debug channel* with this name, but for another line (`E_EXISTS`).

`9` = Command was terminated using `SIGTERM` (`E_INTR`).

`10` = Command receives an unexpected signal (`E_SIGNAL`).

FILES

`/etc/.osm`

`/dev/rtty/dbg`*lineno*

SEE ALSO

`mknod`(2), `stat`(2), `rtty`(7).

System Administrator's Reference Manual

runacct(1M)

NAME

`runacct` - run daily accounting

SYNOPSIS

`/usr/lib/acct/runacct` [*mmdd* [*state*]]

DESCRIPTION

`runacct` is the main daily accounting shell procedure. It is normally initiated via `cron`. `runacct` processes connect, fee, disk, and process accounting files. It also prepares summary files for `prdaily` or billing purposes.

`runacct` takes care not to damage active accounting files or summary files in the event of errors. It records its progress by writing descriptive diagnostic messages into `active`. When an error is detected, a message is written to `/dev/console`, mail [see `mail(1)`] is sent to `root` and `adm`, and `runacct` terminates. `runacct` uses a series of lock files to protect against re-invocation. The files `lock` and `lock1` are used to prevent simultaneous invocation, and `lastdate` is used to prevent more than one invocation per day.

`runacct` breaks its processing into separate, restartable *states* using `statefile` to remember the last *state* completed. It accomplishes this by writing the *state* name into `statefile`. `runacct` then looks in `statefile` to see what it has done and to determine what to process next. *states* are executed in the following order:

`SETUP`
: Move active accounting files into working files.

`WTMPFIX`
: Verify integrity of `wtmp` file, correcting date changes if necessary.

`CONNECT`
: Produce connect session records in binary format.

`PROCESS`
: Convert process accounting records into binary format.

`MERGE`
: Merge the connect and process accounting records.

`FEES` Convert output of `chargefee` into binary format and merge with connect and process accounting records.

`DISK` Merge disk accounting records with connect, process, and fee accounting records.

`MERGETACCT`
: Merge the daily total accounting records in `daytacct` with the summary total accounting records in `/var/adm/acct/sum/tacct`.

`CMS` Produce command summaries.

`USEREXIT`
: Any installation dependent accounting programs can be included here.

`CLEANUP`
: Clean up temporary files and exit.

To restart `runacct` after a failure, first check the `active` file for diagnostics, then fix any corrupted data files, such as `pacct` or `wtmp`. The `lock`, `lock1`, and `lastdate` files must be removed before `runacct` can be restarted. The argument *mmdd* is necessary if `runacct` is being restarted, and specifies the month and day for which `runacct` will rerun the accounting. The entry point for processing is based on the contents of `statefile`; to override this, include the desired *state* on the command line to designate where processing should begin.

EXAMPLES

To start `runacct`:

```
nohup runacct 2> /var/adm/acct/nite/fd2log &
```

To restart `runacct`:

```
nohup runacct 0601 2>> /var/adm/acct/nite/fd2log &
```

To restart `runacct` at a specific *state*:

```
nohup runacct 0601 MERGE 2>> /var/adm/acct/nite/fd2log &
```

NOTES

Normally it is not a good idea to restart `runacct` in the `SETUP` *state*. Run `SETUP` manually and restart via:

`runacct` *mmdd* `WTMPFIX`

If `runacct` failed in the `PROCESS` *state*, remove the last `ptacct` file because it will not be complete.

FILES

`/var/adm/wtmp`
`/var/adm/pacct`*incr*
`/var/adm/acct/nite/active`
`/var/adm/acct/nite/daytacct`
`/var/adm/acct/nite/lock`
`/var/adm/acct/nite/lock1`
`/var/adm/acct/nite/lastdate`
`/var/adm/acct/nite/statefile`

SEE ALSO

`acctcom`(1), `mail`(1), `acct`(1M), `acctcms`(1M), `acctcon`(1M), `acctmerg`(1M), `acctprc`(1M), `acctsh`(1M), `cron`(1M), `fwtmp`(1M), `acct`(2), `utmp`(4).

S02errlogd(1M) - syslogd(1M)

System Administrator's Reference Manual

S02errlogd(1M)
RM600 only

NAME

`S02errlogd` - start/stop the Errlog daemon

SYNOPSIS

`/etc/rc2.d/S02errlogd start | stop`

DESCRIPTION

Three files are used during the Errlog daemon run:

1. Component limit file (CL)

   An important condition for the recording of correctable errors is the availability of a component limit file (CL file) for each hardware component to be monitored. This file contains the component limit parameters (CL parameters) which are valid for a hardware component. The concrete values for the CL parameters can be found in the technical specifications provided by the vendor for the individual hardware components. The CL files must be present in the `/etc/default/diskinfo/errlogd` directory before the Errlog daemon is started.

2. `errlog.cnf` configuration file

   The link between the hardware component and the CL file is established in the `errlog.cnf` file. The `errlog.cnf` file is stored in the `/opt/errlogd` directory.

3. `errlog.sav` export file

   When the Errlog daemon is stopped, its internal data structure is swapped out to this file. The `errlog.sav` file is stored in the `/opt/errlogd` directory.

OPTIONS

`start`

This parameter is used to start the Errlog daemon process. A check is first made to determine whether an Errlog daemon process is already running. If this is the case, the command is terminated with a non-zero return value. If there is no Errlog daemon process already running, the Errlog daemon is started. The `errlog.cnf` file is created, in which the link between the hardware component names and the component limit files (CL files) is established. In this way, each hardware component is assigned a CL file.

a) `errlog.cnf` and `errlog.sav` available

   The configuration parameters in the CL files are combined with parameters already available in the Errlog daemon. The parameters in CL files have priority.

b) `errlog.cnf` available and `errlog.sav` not available

   The configuration parameters are taken from the CL file, and monitoring of the hardware component is restarted.

c) `errlog.cnf` not available and `errlog.sav` available

   The configuration parameters are taken from the `errlog.sav` file. The data context of the Errlog daemon is restored, and monitoring of the hardware component is continued from the point at which it stopped when the Errlog daemon was stopped.

d) `errlog.cnf` and `errlog.sav` not available

   Startup of the Errlog daemon is canceled and terminated with a non-zero return value.

`stop` This parameter is used to stop the Errlog daemon process. In this event, a `SIGHUP` signal is first sent to the Errlog daemon process. The `errlog.cnf` configuration file is deleted. If the Errlog daemon is running in `ALLSAVE` mode, all administration structures (as well as the configuration parameters) of the Errlog daemon are saved in the `errlog.sav` file. The Errlog daemon process then terminates with the `SIGKILL` signal. If the Errlog daemon process is running in `NOSAVE` mode, it terminates immediately with the `SIGKILL` signal.

The `errlog.cnf` file is therefore only available when the Errlog daemon cannot find the appropriate CL files. The return value of the command is 0, and unequal to 0 in the event of an error.

FILES

`/etc/default/diskinfo/errlogd/`*CL-files*

`/opt/errlogd/errlog.cnf`

`/sbin/errlogd`

SEE ALSO

`set_errlogd(1M)`, `errlogd(8)`.

System Administrator's Reference Manual

sac(1M)

NAME

sac - service access controller

SYNOPSIS

sac –t *sanity_interval*

DESCRIPTION

The Service Access Controller (SAC) is the overseer of the server machine. It is started when the server machine enters multiuser mode. The SAC performs several important functions as explained below.

- Customizing the SAC environment

  When sac is invoked, it first checks whether the per-system configuration script /etc/saf/_sysconfig exists [see doconfig(3C)]. sac interprets _sysconfig to customize its own environment. The modifications made to the SAC environment by _sysconfig are inherited by all the children of the SAC. This inherited environment may be modified by the children.

- Starting port monitors

  After it has interpreted the _sysconfig file, the sac reads its administrative file /etc/saf/_sactab. _sactab specifies which port monitors are to be started. For each port monitor to be started, sac forks a child [see fork(2)] and creates a utmp entry with the *type* field set to LOGIN_PROCESS. Each child then interprets its per-port monitor configuration script /etc/saf/*pmtag*/_config, if the file exists. These modifications to the environment affect the port monitor and will be inherited by all its children. Finally, the child process execs the port monitor, using the command found in the _sactab entry. (See sacadm(1M) ; this is the command given with the –c option when the port monitor is added to the system.)

- Polling port monitors to detect failure

  The –t option sets the frequency with which sac polls the port monitors on the system. This time may also be thought of as half of the maximum latency required to detect that a port monitor has failed and that recovery action is necessary.

- Administrative functions

  The Service Access Controller represents the administrative point of control for port monitors. Its administrative tasks are explained below.

When queried (sacadm with either –l or –L), the Service Access Controller returns the status of the port monitors specified, which sacadm prints on the standard output. A port monitor may be in one of six states:

ENABLED
: The port monitor is currently running and is accepting connections. See sacadm(1M) with the –e option.

DISABLED
: The port monitor is currently running and is not accepting connections. See sacadm with the –d option, and see NOTRUNNING, below.

STARTING
: The port monitor is in the process of starting up. STARTING is an intermediate state on the way to ENABLED or DISABLED.

FAILED
: The port monitor was unable to start and remain running.

STOPPING
: The port monitor has been manually terminated but has not completed its shutdown procedure. STOPPING is an intermediate state on the way to NOTRUNNING.

NOTRUNNING
: The port monitor is not currently running. (See sacadm(1M) with –k.) This is the normal "not running"

state. When a port monitor is killed, all ports it was monitoring are inaccessible. It is not possible for an external user to tell whether a port is not being monitored or the system is down. If the port monitor is not killed but is in the `DISABLED` state, it may be possible (depending on the port monitor being used) to write a message on the inaccessible port telling the user who is trying to access the port that it is disabled. This is the advantage of having a `DISABLED` state as well as the `NOTRUNNING` state.

When a port monitor terminates, the SAC removes the `utmp` entry for that port monitor.

The SAC receives all requests to enable, disable, start, or stop port monitors and takes the appropriate action.

The SAC is responsible for restarting port monitors that terminate. Whether or not the SAC will restart a given port monitor depends on two things:

- the restart count specified for the port monitor when the port monitor was added by `sacadm`; this information is included in `/etc/saf/`*pmtag*`/_sactab`;
- the number of times the port monitor has already been restarted.

FILES

```
/etc/saf/_sactab
/etc/saf/_sysconfig
/var/saf/_log
/var/adm/utmp
```

SEE ALSO

`pmadm`(1M), `sacadm`(1M), `doconfig`(3C).

System Administrator's Reference Manual

sacadm(1M)

NAME

`sacadm -` service access controller administration

SYNOPSIS

`sacadm –a –p` *pmtag* `–t` *type* `–c` *cmd* `–v`*ver* [`–f dx`] [`–n` *count*] [`–y` *comment*] [`–z` *script*]

`sacadm –r –p` *pmtag*

`sacadm –s –p` *pmtag*

`sacadm –k –p` *pmtag*

`sacadm –e –p` *pmtag*

`sacadm –d –p` *pmtag*

`sacadm –l` [`–p` *pmtag* | `–t` *type*]

`sacadm –L` [`–p` *pmtag* | `–t` *type*]

`sacadm –g –p` *pmtag* [`–z` *script*]

`sacadm –G` [`–z` *script*]

`sacadm –x` [`–p` *pmtag*]

DESCRIPTION

`sacadm` is the administrative command for the upper level of the Service Access Facility hierarchy, that is, for port monitor administration. `sacadm` performs the following functions:

- adds or removes a port monitor
- starts or stops a port monitor
- enables or disables a port monitor
- installs or replaces a per-system configuration script
- installs or replaces a per-port monitor configuration script
- prints requested port monitor information

Requests about the status of port monitors (`–l` and `–L`) and requests to print per-port monitor and per-system configuration scripts (`–g` and `–G` without the `–z` option) may be executed by any user on the system. Other `sacadm` commands may be executed only by a privileged user.

OPTIONS

`–a` Adds a port monitor. When adding a port monitor, `sacadm` creates the supporting directory structure in `/etc/saf` and `/var\/saf` and adds an entry for the new port monitor to `/etc/saf/_sactab`. The file `_sactab` already exists on the delivered system. Initially, it is empty except for a single line, which contains the version number of the Service Access Controller.

Unless the command line that adds the new port monitor includes a `–f` option with the argument `x`, the new port monitor will be started. Because of the complexity of the options and arguments that follow the `–a` option, it may be convenient to use a command script or the menu system to add port monitors. If you use the menu system, enter "`sysadm ports`" and then choose the "`port_monitors`" option.

`–c` *cmd*

Executes the command string *cmd* to start a port monitor. The `–c` option may be used only with a `–a`. A `–a` option requires a `–c`.

`–d` Disables the port monitor *pmtag*.

`–e` Enables the port monitor *pmtag*.

`–f dx`

The –f option specifies one or both of the following two flags which are then included in the flags field of the _sactab entry for the new port monitor. If the –f option is not included on the command line, no flags are set and the default conditions prevail. By default, a port monitor is started. A –f option with no following argument is illegal.

d Do not enable the new port monitor.

x Do not start the new port monitor.

–g The –g option is used to request output or to install or replace the per-port monitor configuration script /etc/saf/*pmtag*/_config. –g requires a –p option. The –g option with only a –p option prints the per-port monitor configuration script for port monitor *pmtag*. The –g option with a –p option and a –z option installs the file *script* as the per-port monitor configuration script for port monitor *pmtag*. Other combinations of options with –g are invalid.

–G The –G option is used to request output or to install or replace the per-system configuration script /etc/saf/_sysconfig. The –G option by itself prints the per-system configuration script. The –G option in combination with a –z option installs the file *script* as the per-system configuration script. Other combinations of options with a –G option are invalid.

–k Stops a port monitor *pmtag*.

–l The –l option is used to request port monitor information. The –l by itself lists all port monitors on the system. The –l option in combination with the –p option lists only the port monitor specified by *pmtag*. A –l in combination with the –t option lists all port monitors of type *type*. Any other combination of options with the –l option is invalid.

–L The –L option is identical to the –l option except that the output appears in a condensed format.

–n *count*
Sets the restart count to *count*. If a restart count is not specified, count is set to 0. A count of 0 indicates that the port monitor is not to be restarted if it fails.

–p *pmtag*
Specifies the tag associated with a port monitor.

–r Removes a port monitor *pmtag*. sacadm removes the port monitor entry from /etc/saf/_sactab. If the removed port monitor is not running, then no further action is taken. If the removed port monitor is running, the Service Access Controller (SAC) sends it SIGTERM to indicate that it should shut down. Note that the port monitor's directory structure remains intact.

–s Starts a port monitor. The SAC starts the port monitor *pmtag*.

–t *type*
Specifies the port monitor type.

–v *ver*
Specifies the version number of the port monitor. This version number may be given as

–v `*pmspec* –V`

where *pmspec* is the special administrative command for port monitor *pmtag*. This special command is ttyadm for ttymon and nlsadmin for listen. The version stamp of the port monitor is known by the command and is returned when *pmspec* is invoked with a –V option.

–x The –x option by itself tells the SAC to read its database file (_sactab). The –x option with the –p option tells port monitor *pmtag* to read its administrative file.

–y *comment*
Includes *comment* in the _sactab entry for port monitor *pmtag*.

–z *script*
Used with the –g and –G options to specify the name of a file that contains a configuration script. With the –g option, *script* is a per-port monitor configuration script; with –G it is a per-system configuration script. Modifying a configuration script is a three-step procedure. First a copy of the existing script is made (–g or –G). Then the copy is edited. Finally, the copy is put in place over the existing script (–g or

–G with –z).

DIAGNOSTICS

If successful, sacadm will exit with a status of 0. If sacadm fails for any reason, it will exit with a nonzero status. Options that request information will write the information on the standard output. In the condensed format (–L), port monitor information is printed as a sequence of colon-separated fields; empty fields are indicated by two successive colons. The standard format (–l) prints a header identifying the columns, and port monitor information is aligned under the appropriate headings. In this format, an empty field is indicated by a hyphen. The comment character is #.

EXAMPLES

The following command line adds a port monitor. The port monitor tag is npack; its type is listen; if necessary, it will restart three times before failing; its administrative command is nlsadmin; and the configuration script to be read is in the file script:

```
sacadm –a –p npack –t listen –c /usr/lib/saf/listen npack\
–v `nlsadmin –V` –n 3 –z script
```

Remove a port monitor whose tag is pmtag:

```
sacadm –r –p pmtag
```

Start the port monitor whose tag is pmtag:

```
sacadm –s –p pmtag
```

Stop the port monitor whose tag is pmtag:

```
sacadm –k –p pmtag
```

Enable the port monitor whose tag is pmtag:

```
sacadm –e –p pmtag
```

Disable the port monitor whose tag is pmtag:

```
sacadm –d –p pmtag
```

List status information for all port monitors:

```
sacadm –l
```

List status information for the port monitor whose tag is pmtag:

```
sacadm –l –p pmtag
```

List the same information in condensed format:

```
sacadm –L –p pmtag
```

List status information for all port monitors whose type is listen:

```
sacadm –l –t listen
```

Replace the per-port monitor configuration script associated with the port monitor whose tag is pmtag with the contents of the file file.config:

```
sacadm –g –p pmtag –z file.config
```

LOCALE

The LC_MESSAGES environment variable governs the language in which message texts are displayed.

If LC_MESSAGES is undefined or is defined as the null string, it defaults to the value of LANG. If LANG is likewise undefined or null, the system acts as if it were not internationalized.

If any of the locale variables has an invalid value, the system acts as if none of the variables was set.

The LC_ALL environment variable governs the entire locale. LC_ALL takes precedence over all the other environment variables that affect internationalization.

FILES

/etc/saf/_sactab
/etc/saf/_sysconfig
/etc/saf/*pmtag*/_config

SEE ALSO

nlsadmin(1M), listen(1M), pmadm(1M), sac(1M), ttyadm(1M), ttymon(1M), doconfig(3C).

System Administrator's Reference Manual

sar(1M)

NAME

sar:

sa1, sa2, sadc - system activity report package

SYNOPSIS

/usr/lib/sa/sadc [*t n*] [*ofile*]

/usr/lib/sa/sa1 [*t n*]

/usr/lib/sa/sa2 [*options*] [–s *time*] [–e *time*] [–i *sec*]

DESCRIPTION

System activity data can be accessed at the special request of a user [see sar(1)] and automatically, on a routine basis, as described here. The operating system contains several counters that are incremented as various system actions occur. These include counters for CPU utilization, buffer usage, disk and tape I/O activity, TTY device activity, switching and system-call activity, file access, queue activity, interprocess communications, paging, and Remote File Sharing.

sadc and two shell procedures, sa1 and sa2, are used to sample, save, and process this data.

sadc, the data collector, samples system data *n* times, with an interval of *t* seconds between samples, and writes in binary format to *ofile* or to standard output. The sampling interval *t* should be greater than 5 seconds; otherwise, the activity of sadc itself may affect the sample. If *t* and *n* are omitted, a special record is written. This facility is used at system boot time, when booting to a multiuser state, to mark the time at which the counters restart from zero. For example, the /etc/init.d/perf file writes the restart mark to the daily data by the command entry:

```
su sys –c "/usr/lib/sa/sadc /var/adm/sa/sa`date +%d`"
```

The shell script sa1, a variant of sadc, is used to collect and store data in the binary file /var/adm/sa/sa*dd*, where *dd* is the current day. The arguments *t* and *n* cause records to be written *n* times at an interval of *t* seconds, or once if omitted. The following entries in /var/spool/cron/crontabs/sys will produce records every 20 minutes during working hours and hourly otherwise:

```
0 * * * 0-6 /usr/lib/sa/sa1
20,40 8–17 * * 1–5 /usr/lib/sa/sa1
```

See crontab(1) for details.

The shell script sa2, a variant of sar, writes a daily report in the file /var/adm/sa/sar*dd*. The options are explained in sar(1) . The following entry in /var/spool/cron/crontabs/sys will report important activities hourly during the working day:

```
5 18 * * 1–5 /usr/lib/sa/sa2 –s 8:00 –e 18:01 –i 1200 –A
```

The structure of the binary daily data file is:

```
struct sa {
  /* all per-cpu statistics */
  struct  sysinfo si[NCPU];   /* defined in /usr/include/sys/sysinfo.h */
  struct  minfo   mi[NCPU];   /* defined in /usr/include/sys/sysinfo.h */
  struct  vminfo vmi[NCPU];
  struct  ufs_statistics ufs[NCPU];     /* new mips ufs-statistics */
  rf_srv_info_t   rf_srv;
  fsinfo_t        rfs_in;
  fsinfo_t        rfs_out;
  rfc_info_t      rfc;
  struct kmeminfo kmi[NCPU];
  int     minserve;
  int     maxserve;
  ssize_t szinode;   /* current size of inode table */
```

```
  uint    szfile;    /* current size of file table */
  uint    szproc;    /* current size of proc table */
  int     szlckf;    /* current size of file record header table */
  long    szlckr;    /* current size of file record lock table */
  ssize_t mszinode;  /* maximum size of inode table */
  int     mszfile;   /* maximum size of file table */
  int     mszproc;   /* maximum size of proc table */
  int     mszlckf;   /* maximum size of file record header table */
  int     mszlckr;   /* maximum size of file record lock table */
  long    inodeovf;  /* cumulative overflows of inode table since boot */
  long    fileovf;   /* cumulative overflows of file table since boot */
  long    procovf;   /* cumulative overflows of proc table since boot */
  time_t  ts;        /* time stamp */
  int apstate;
  struct shlbinfo shlb;           /* shared lib info */
  struct rtminfo  rt;
  struct syswait swt;
  struct dev_sar *devio; /* device unit information */
  int     deviosz;       /* size of device unit information structure */
#define IO_OPS  0  /* number of I /O requests since boot */
#define IO_BCNT 1  /* number of blocks transferred since boot */
#define IO_ACT  2  /* cumulative time in ticks when drive is active */
#define IO_RESP 3  /* cumulative I/O response time in ticks since boot */
};
```

NOTES

The old `struct kmeminfo km;` (SINIX-N only) definition still applies for single-processor systems.

FILES

`/var/adm/sa/sa`*dd*
: daily data file

`/var/adm/sa/sar`*dd*
: daily report file

`/tmp/sa.adrfl`
: address file

SEE ALSO

`crontab`(1), `sar`(1), `timex`(1).

System Administrator's Reference Manual

savelog(1M)

NAME

`savelog` - save a log file into an archive directory

SYNOPSIS

`/usr/sbin/savelog` [`-d` *savedir*] [`-c` *cycles*] [`-n` *noncomp*] [`-s`] [`-o` *owner*] [`-g` *group*] [`-m` *mode*] [*logfile*]

DESCRIPTION

`savelog` is used to keep the size of a log file down by periodically truncating it. In order not to lose information, several versions may be kept in an archive directory. To save on space, one may arrange to keep these archived log files in compressed form.

OPTIONS

`-d` *savedir*
: The directory used to store archived versions of *logfile*. *savedir* should be a full pathname or a pathname relative to *logfile*.

  If *savedir* does not exist it will be created. The default path is `./.old`.

`-c` *cycles*
: The number of archived versions of *logfile* to save. The default is 5.

`-n` *noncomp*
: The first *noncomp* archives are not compressed. If *noncomp* is 0 then all archived files will be compressed. If *noncomp* is the same as *cycles* then no archived files will be compressed. The default is 1.

`-s`
: Process *logfile* even if it is empty.

`-o` *owner*
: Ensures that *logfile* and related archives are owned by *owner*. If *savedir* needs to be created then it will also be owned by *owner*. Existing archive directories are not affected.

`-g` *group*
: Ensures that *logfile* and related archives are owned by *group*. If *savedir* needs to be created then it will also be owned by *group*. Existing archive directories are not affected.

`-m` *mode*
: Ensures that *logfile* and related archives have mode *mode*. If *logfile* was specified then *mode* does not apply to the *savedir* archive directory.

logfile
: The full pathname of the file to be archived. If no *logfile* is specified then `savelog` will only attempt to create the *savedir* directory.

SEE ALSO

`chgrp`(1), `chmod`(1), `chown`(1), `compress`(1), `cron`(1M).

System Administrator's Reference Manual

sdb(1M)

NAME

sdb:
sdb_insert, sdb_delete, sdb_replace, sdb_obj, sdb_attr, sdb_check - tools for accessing the sdb.daps software database

SYNOPSIS

sdb_insert [–r "*attribute_value*"...] [–V *sep*] [–h]

sdb_delete [–s "*selection*"...] [–V *sep*] [–h]

sdb_replace [–r "*attribute_value*"...] [–V *sep*] [–h]

sdb_obj [–n *node_list*] [–s "*selection*"...] [–a] [–V *sep*] [–h]

sdb_attr [–n *node*] [–s "*selection*"...] [–V *sep*] [–h]

sdb_check [–n *node_list*] [–C *cluster*] –s "*selection*"... [–s "*selection*"...] [–V *sep*] [–h]

DESCRIPTION

The sdb_insert command can be used to insert new data records in the local software database. The complete package name, comprising the attributes PkgName, PkgVersion and PkgLoad must always be entered. This simply serves to document the existence of the package. If an object class and an attribute name are entered, their existence is also documented. If several object classes and/or attributes are entered, several data records are required in the software database.

The sdb_delete command can be used to delete data records from the local software database. All data records are always deleted, where all predefined selection criteria do not match fully. If the package name is predefined, i.e. simply the PkgName attribute, all data records for this package are deleted including all versions and loads.

The sdb_replace is used to insert new data records in the local software database or to modify the AttrValue attribute. All data records are deleted first, in cases where all attributes (apart from AttrValue) match the predefined data record. The new data record (possibly including the AttrValue attribute) is then re-entered in the software database. This guarantees the uniqueness of the data records in the software database.

The sdb_obj command is used to define a list of object classes for the specified list of node names – or for the entire CADMIN cluster if no node was predefined – which can at least be found on one of the specified nodes and at the same time fulfill the predefined selection criteria. Valid selection criteria would be a package name or module name. The list of names of object classes, which exist on at least one node in the cluster or in the specified list of nodes, is sent to standard output (one object class name per line).

If the –a option was specified, the list of object classes located on all specified nodes is defined. The result is only not 0, if internal errors occurred during data access, but not if no object class was found on all nodes with the required features.

The sdb_attr command allows the attribute value to be specified for a predefined node and a predefined attribute name. An object class must be specified. Precisely one node can be predefined. The local system is the default node.

The sdb_check command can be used to check which of the nodes in a predefined list of node names in a predefined cluster meet all selection criteria. A list of node names that fulfill all selection criteria is returned. The check is always only performed locally. At least one selection criterion must be predefined for an object class.

If no cluster is transferred explicitly with the –C option, the databases of the predefined nodes are checked locally. If there is no node specified, only the current (= local) node is checked.

If a cluster is specified explicitly, the databases of all nodes in the cluster are checked locally (those maintained up to date locally by the cluster operating system).

If a list of nodes is transferred in addition to the explicitly specified cluster, only these will be checked. If a cluster is predefined explicitly, error messages may arise if the current node or one of the predefined nodes is not a member of the specified cluster.

If selection criteria regarding different object classes are to be checked, the returned list can be transferred again to the `sdb_check` command.

OPTIONS

`-n` *node*
: The *node* argument defines the software database to be changed, since the node name is a component of the path name `/var/soft-DB/`*node*`/sdb.daps`. The default value for *node* is the local system name (`uname -n`).

`-n` *node_list*
: This is a list of system names separated by commas (`uname -n`).

`-C` *cluster*
: Name of the cluster in which the specified selection criteria are to be checked for all members or for the predefined list of nodes.

`-r "`*attribute_value*`"...`
: The new attribute values are specified as a list of expressions in the form *attributename*=*value*, where *attributename* is one of the known attribute names [see `sdb.def(4)` : `PkgName`, `PkgVersion`, `PkgLoad`, `Subject`, `ObjClass`, `AttrName`, `AttrValue`] and *value* is the new attribute value. Several such expressions are separated by a freely definable separator.

  The "=" operator can be surrounded by any number of blanks and tabs. The entire list must be combined to form a single argument with "...".

  Several `-r` options can be used.

  Example:

```
-r "PkgName = SIlog3 @ PkgVersion = 3.0A10 @ PkgLoad = 00 @ ObjClass =
logentries @ AttrName = SORT" -r "AttrValue = reverse order"
```

`-s "`*selection*`"...`
: The selection criteria are specified as a list of expressions in the form *attributename*==*value*, where *attributename* is one of the known attributes names [see `sdb.def(4)` : `PkgName`, `PkgVersion`, `PkgLoad`, `Subject`, `ObjClass`, `AttrName`, `AttrValue`] and *value* is the attribute value. Several such expressions are separated by a freely definable separator. The only comparative operator permitted is "==".

  The "=" operator can be surrounded by any number of blanks and tabs. The entire list must be combined to form a single argument with "...".

  Several `-s` options can be used.

  Example:

```
-s "PkgName==SIlog3 @ PkgVersion == 3.0A20 @ PkgLoad == 0" -s "ObjClass ==
logentries @ AttrName==SORT @ AttrValue == reverse order"
```

`-a`
: Only for `sdb_obj`: The object classes must not only exist on one node, but on all selected nodes.

`-V` *sep*
: Defines a specific separator. Only a single character is permitted. If this character appears in the readable ASCII data, it must be masked in both input and output with "\". The default separator is `@`.

`-h`
: Usage for this command.

DIAGNOSTICS

The return value is 0 if no error is detected. If one of the `sdb` commands discovers an error situation, this is reported to standard error output and the return value is not 0.

`1` `"Illegal/less options!"`

Bad call: Incorrect command options or option values were used.

2 "Illegal package name (>16)!"

The package of the specified name cannot be found.

3 "Illegal package version!"

The specified package version is more than 8 characters long.

4 sdb_check: "No nodes to check!"

You have not specified a list of nodes to be checked and the list of nodes belonging to the cluster is also blank.

sdb_insert, sdb_delete, sdb_replace: "You are not authorised to administrate sdb!"

You are not authorized to modify the software database.

5 "Illegal/no package load!" You have not specified a package load.

6 sdb_attr, sdb_delete: "Illegal/no object class name!"

You have not specified an object class name.

sdb_insert, sdb_replace: "Attribute name without object class name!"

You have predefined an attribute name without specifying the object class to which it belongs.

7 sdb_attr: "Illegal/no attribute name!"

You have not specified an attribute name.

sdb_insert, sdb_replace, sdb_obj: "Attribute value without attribute name!"

You have predefined an attribute value without specifying the attribute name to which it is to be assigned.

8 "Subject too long (>16)!"

The module/product name you specified is too long.

9 "Object class name too long (>32)!"

The object class name you specified is too long.

10 "Attribute name too long (>32)!"

The attribute name you specified is too long.

11 "Attribute value too long (>380)!"

The attribute value you specified is too long.

99 "sdb_check: "No given node meets the selection criteria!"

None of the specified nodes fulfills the selection criteria.

NOTES

The attribute values are not checked for admissibility, plausibility or completeness. In particular, the existence and correctness of the package name, versions and loads is not checked.

It is not checked whether the attribute names Subjects, ObjClass, AttrName and AttrValue are also actually made available by the package. The entries in the software database are used to publish the available features of the package, which may be queried by any user. These features generally affect the exchange of data with the package using the GET/SET commands and can therefore generally be assigned to the object classes and their attributes. However, there are also features that affect the quality of the data exchange (e.g. value ranges, sorting, etc.). New object and attribute names can be agreed in this case.

A package must be able to guarantee consistent and permanent usage of names and entries throughout its life cycle.

FILES

/var/soft-DB/*node*/sdb.def

`/var/soft-DB/`*node*`/sdb.daps`

SEE ALSO

`sdb.def`(4).

System Administrator's Reference Manual

secure(1M)

NAME

`secure` - check security

SYNOPSIS

`secure` [*options*]

DESCRIPTION

The `secure` program tries to find out security problems of a Reliant UNIX system and displays potential security risks on the screen.

The functional scope of `secure` depends on whether a non-privileged user or the system administrator calls the command.

If the `secure` command is called by a non-privileged user, the information is only output for the login name of the user executing the call. A call by the system administrator checks the entire computer system.

`secure` offers the following functional scope to the non-privileged user:

- It lists all files for which the set-user-id or set-group-id bit is set. These files constitute a security risk, since anyone who processes such a file has the permissions of the owner, with which he/she can perform any actions.
- It lists all directories in the user's file tree for which all system users have write permission.
- It searches for programs with the name `su`.
- It analyzes the shell variables. If there is no `.profile` file for the login name being checked, it uses the `/etc/profile` file.

`secure` offers an extended functional range for the system administrator. The following information is displayed:

- login names for which there is no password
- "pseudo login names" for which an invalid password is specified, and which therefore do not permit `login`
- login names with a double user ID
- login names for which no login directory is defined
- login names which have not been used in the last 30 days
- files with the set-user-id or set-group-id bit set. If these are text files (i.e. possibly shell scripts), an additional warning is output
- directories for which all system users have write permission
- files in the `/dev` directory which are not special files
- special files in the `/dev` directory for which all system users have write permission
- special files which are not in the `/dev` directory
- copies of `/bin/sh`
- programs with the name `su`
- escape sequences in the mailbox

OPTIONS

`-h` Displays help texts.

`-l` `secures` attempts to determine the password for the login name in order to establish whether the password assigned is too simple.

For this purpose, `secure` contains an internal password list. It searches for the user's surname, forename, company name etc.

If this option is called by the system administrator, `secure` examines all login names. This process can then take up a lot of time.

`-s` Status of an already running `secure -l`. You get the user whose password `secure -l` actually tries to encrypt and the percentage of users checked so far.

The `-s` option is available to the system administrator only.

`-t` The `-t` option enables the system administrator to use `secure` to analyze any login name in the status of a non-privileged user.

Once `secure -t` has been called, the system administrator is requested to enter a password. With the help of this password, he/she can then log in to any login name via `login` and can analyze this name in the status of a non-privileged user with `secure`.

The `-t` option is available to the system administrator only.

FILES

Temporary files which are created while `secure` is running:

```
/dev/tmp/sec_st
/dev/tmp/no_dev
/dev/tmp/out_dev
/dev/tmp/s_bit
/dev/tmp/t_bit
/dev/tmp/w_dev
/dev/tmp/sh_cp
/dev/tmp/f_su
/dev/tmp/mail_e
```

SEE ALSO

`chmod(1)`, `login(1)`, `passwd(1)`, `su(1)`.

System Administrator's Reference Manual

sendmail(1M)

NAME

`sendmail` - send mail over the internet

SYNOPSIS

`sendmail` [*options*] *address ...*

DESCRIPTION

`sendmail` sends a message to one or more recipients, routing the message over whatever networks are necessary. `sendmail` forwards the message to the correct place via the network.

`sendmail` is not intended as a user interface routine; other programs provide user-friendly front ends; `sendmail` is used only to deliver pre-formatted messages.

With no options, `sendmail` reads its standard input up to an end-of-file or a line consisting only of a single dot and sends a copy of the message found there to all of the addresses listed. It determines the network(s) to use based on the syntax and contents of the addresses.

Local addresses are looked up in a file and aliased appropriately. Aliasing can be prevented by preceding the address with a backslash. Normally the sender is not included in any alias expansions, e.g. if "john" sends to "group", and "group" includes "john" in the expansion, then the letter will not be delivered to "john".

OPTIONS

`–B` *type*
: Set the body type to *type*. Current legal values `7BIT` or `8BITMIME`.

`–ba` Switch to ARPANET mode. All input lines must end with a CR-LF, and all messages will be generated with a CR-LF at the end. Also, the "From:" and "Sender:" fields are examined for the name of the sender.

`–bd` Run as a daemon. This requires Berkeley IPC. `sendmail` will fork and run in background listening on socket 25 for incoming SMTP connections. This is normally run from `/etc/rc/rc2.d`.

`–bD` Same as `–bd` except runs in foreground.

`–bh` Print the persistent host status database.

`–bH` Purge the persistent host status database.

`–bi` Initialize the alias database.

`–bm` Deliver mail in the usual way (default).

`–bp` Print a listing of the queue.

`–bs` Use the SMTP protocol as described in RFC821 on standard input and output. This option implies all the operations of the `–ba` option that are compatible with SMTP.

`–bt` Run in address test mode. This mode reads addresses and shows the steps in parsing; it is used for debugging configuration tables.

`–bv` Verify names only – does not try to collect or deliver a message. Verify mode is normally used for validating users or mailing lists.

`–C`*file*
: Use alternate configuration file. `sendmail` refuses to run as root if an alternate configuration file is specified.

`–d`*X* Set debugging value to *X*.

`–F` *fullname*
: Set the full name of the sender.

`–f` *name*

Set the name of the "from" person (i.e. the sender of the mail).

`-h` *N* Set the hop count to *N*. The hop count is incremented every time the mail is processed. When it reaches a limit, the mail is returned with an error message, the victim of an aliasing loop. If not specified, "Received:" lines in the message are counted.

`-i` Ignore dots alone on lines by themselves in incoming messages. This should be set if you are reading data from a file.

`-N` *dsn*
Set delivery status notification conditions to *dsn*, which can be `never` for no notifications or a comma separated list of the values `failure` to be notified if delivery failed, `delay` to be notified if delivery is delayed, and `success` to be notified when the message is successfully delivered.

`-n` Don't do aliasing.

`-O` *option*=*value*
Set option *option* to the specified *value*. This form uses long names. See below for more details.

`-o`*x**value*
Set option *x* to the specified *value*. This form uses single character names only. The short names are not described in this manual page; see the "Sendmail Installation and Operation Guide" (= `/opt/sendmail_doc/op.ps`) for details.

`-p`*protocol*
Sets the name of the *protocol* used to receive the message. This can be a simple protocol name such as "UUCP" or a protocol and hostname, such as "UUCP:ucbvax".

`-q`[*time*]
Processed saved messages in the queue at given intervals. If *time* is omitted, process the queue once. *time* is given as a tagged number, with `s` being seconds, `m` being minutes, `h` being hours, `d` being days, and `w` being weeks. For example, `-q1h30m` or `-q90m` would both set the timeout to one hour thirty minutes.

If *time* is specified, `sendmail` will run in background. This option can be used safely with `-bd`.

`-qI`*substr*
Limit processed jobs to those containing *substr* as a substring of the queue id.

`-qR`*substr*
Limit processed jobs to those containing *substr* as a substring of one of the recipients.

`-qS`*substr*
Limit processed jobs to those containing *substr* as a substring of the sender.

`-R` *return*
Set the amount of the message to be returned if the message bounces. The *return* parameter can be `full` to return the entire message or `hdrs` to return only the headers.

`-r`*name*
An alternate and obsolete form of the `-f` option.

`-t` Reads message for recipients. "To:", "Cc:", and "Bcc:" lines will be scanned for recipient addresses. The "Bcc:" line will be deleted before transmission. Any addresses in the argument list will be suppressed, that is, they will not receive copies even if listed in the message header.

`-U` Initial (user) submission. This should *always* be set when called from a user agent such as `Mail` or `exmh` and *never* be set when called by a network delivery agent such as `rmail`.

`-V` *envid*
Set the original envelope id. This is propagated across SMTP to servers that support DSNs and is returned in DSN-compliant error messages.

`-v` Goes into verbose mode. Alias expansions will be announced, etc.

`-X`*logfile*
Logs all traffic in and out of mailers in the indicated log file. This should only be used as a last resort for

debugging mailer bugs. It will log a lot of data very quickly.

Processing options

There are also a number of processing options that may be set. Normally these will only be used by a system administrator. Options may be set either on the command line using the `-o` option (for short names), the `-O` option (for long names), or in the configuration file. This is a partial list limited to those options that are likely to be useful on the command line and only shows the long names; for a complete list (and details), consult the "Sendmail Installation and Operation Guide" (= `/opt/sendmail_doc/op.ps`).

The options are:

`AliasFile=`*file*
: Use alternate alias file.

`HoldExpensive`
: On mailers that are considered "expensive" to connect to, don't initiate immediate connection. This requires queueing.

`CheckpointInterval=`*N*
: Checkpoint the queue file after every *N* successful deliveries (default 10). This avoids excessive duplicate deliveries when sending to long mailing lists interrupted by system crashes.

`DeliveryMode=`*x*
: Set the delivery mode to *x*. Delivery modes are:

  `i` for interactive (synchronous) delivery,

  `b` for background (asynchronous) delivery,

  `q` for queue only – i.e., actual delivery is done the next time the queue is run, and

  `d` for deferred – the same as

  `q` except that database lookups (notably DNS and NIS lookups) are avoided.

`ErrorMode=`*x*
: Set error processing to mode *x*. Valid modes are:

  `m` to mail back the error message,

  `w` to "write" back the error message (or mail it back if the sender is not logged in),

  `p` to print the errors on the terminal (default),

  `q` to throw away error messages (only exit status is returned), and

  `e` to do special processing for the BerkNet.

  If the text of the message is not mailed back by modes `m` or `w` and if the sender is local to this machine, a copy of the message is appended to the file `dead.letter` in the sender's home directory.

`SaveFromLine`
: Save UNIX style "From" lines at the front of messages.

`MaxHopCount=`*N*
: The maximum number of times a message is allowed to "hop" before we decide it is in a loop.

`IgnoreDots`
: Do not take dots on a line by themselves as a message terminator.

`SendMimeErrors`
: Send error messages in MIME format. If not set, the DSN (Delivery Status Notification) SMTP extension is disabled.

`ConnectionCacheTimeout=`*timeout*
: Set connection cache timeout.

`ConnectionCacheSize=`*N*
: Set connection cache size.

`LogLevel=`*n*
: The log level.

`MeToo`
Send to "me" (the sender) also if I am in an alias expansion.

`CheckAliases`
Validate the right hand side of aliases during a `newaliases(1M)` command.

`OldStyleHeaders`
If set, this message may have old style headers. If not set, this message is guaranteed to have new style headers (i.e., commas instead of spaces between addresses). If set, an adaptive algorithm is used that will correctly determine the header format in most cases.

`QueueDirectory=`*queuedir*
Select the directory in which to queue messages.

`StatusFile=`*file*
Save statistics in the named file.

`Timeout.queuereturn=`*time*
Set the timeout on undelivered messages in the queue to the specified time. After delivery has failed (e.g., because of a host being down) for this amount of time, failed messages will be returned to the sender. The default is five days.

`UserDatabaseSpec=`*userdatabase*
If set, a user database is consulted to get forwarding information. You can consider this an adjunct to the aliasing mechanism, except that the database is intended to be distributed; aliases are local to a particular host. This may not be available if your sendmail does not have the `–USERDB` option compiled in.

`ForkEachJob`
Fork each job during queue runs. May be convenient on memory-poor machines.

`SevenBitInput`
Strip incoming messages to seven bits.

`EightBitMode=`*mode*
Set the handling of eight bit input to seven bit destinations to *mode*:

`m` (mimefy) will convert to seven-bit MIME format,

`p` (pass) will pass it as eight bits (but violates protocols), and

`s` (strict) will bounce the message.

`MinQueueAge=`*timeout*
Sets how long a job must ferment in the queue between attempts to send it.

`DefaultCharSet=`*charset*
Sets the default character set used to label 8-bit data that is not otherwise labelled.

`DialDelay=`*sleeptime*
If opening a connection fails, sleep for *sleeptime* seconds and try again. Useful on dial-on-demand sites.

`NoRecipientAction=`*action*
Set the behavior when there are no recipient headers ("To:", "Cc": or "Bcc:") in the message to *action*:

`none` leaves the message unchanged,

`add-to`
adds a "To:" header with the envelope recipients,

`add-apparently-to`
adds an "Apparently-To:" header with the envelope recipients,

`add-bcc`
adds an empty "Bcc:" header, and

`add-to-undisclosed`
adds a header reading "To: undisclosed-recipients:;".

`MaxDaemonChildren=`*N*

Sets the maximum number of children that an incoming SMTP daemon will allow to spawn at any time to *N*.

`ConnectionRateThrottle=`*N*
Sets the maximum number of connections per second to the SMTP port to *N*.

In aliases, the first character of a name may be a vertical bar to cause interpretation of the rest of the name as a command to pipe the mail to. It may be necessary to quote the name to keep `sendmail` from suppressing the blanks from between arguments. For example, a common alias is:

```
msgs: "|/usr/bin/msgs -s"
```

Aliases may also have the syntax `:include:`*filename* to ask `sendmail` to read the named file for a list of recipients. For example, an alias such as:

```
poets: ":include:/usr/local/lib/poets.list"
```

would read `/usr/local/lib/poets.list` for the list of addresses making up the group.

DIAGNOSTICS

`sendmail` returns an exit status describing what it did. The codes are defined in `<sysexits.h>`:

`00 (EX_OK)`
Successful completion on all addresses.

`67 (EX_NOUSER)`
User name not recognized.

`69 (EX_UNAVAILABLE)`
Catchall meaning necessary resources were not available.

`70 (EX_SOFTWARE)`
Internal software error, including bad arguments.

`71 (EX_OSERR)`
Temporary operating system error, such as "cannot fork".

`68 (EX_NOHOST)`
Host name not recognized.

`75 (EX_TEMPFAIL)`
Message could not be sent immediately, but was queued.

If invoked as `newaliases`, `sendmail` will rebuild the alias database. If invoked as `mailq`, `sendmail` will print the contents of the mail queue.

FILES

Except for the file `/etc/sendmail.cf` itself, the following pathnames are all specified in `/etc/sendmail.cf`. Thus, these values are only approximations.

`/etc/aliases`
Raw data for alias names

`/etc/aliases.db`
Data base of alias names

`/etc/sendmail.cf`
Configuration file

`/usr/lib/sendmail.hf`
Help file

`/etc/sendmail.st`
Collected statistics

`/var/spool/mqueue/*`
Temporary files

`/etc/sendmail.pid`
The process id of the daemon

`/opt/sendmail_doc/op.ps`

Sendmail Installation and Operation Guide; PostScript format

`/opt/sendmail_doc/intro.ps`

Sendmail – An Internetwork Mail Router; PostScript format

SEE ALSO

`mail`(1-ucb), `rmail`(1-ucb), `vacation`(1-ucb), `mail.local`(1M), `mailq`(1M), `newaliases`(1M), `syslog`(3C), `aliases`(4).

System Administrator's Reference Manual

set_errlogd(1M)
RM600 only

NAME

System Administrator's Reference Manual

set_lar(1M)
RM600 only

NAME

`set_lar -` deactivate controller/set attributes

SYNOPSIS

`set_lar` `–c` *cabinet_ID* `–b` *bus_ID* [`–s` *slot_ID*] [`–n` *channel*] [`–A` *attr*] [`–i` *id*] [`–e` *err*] [`–m` *comment*]

DESCRIPTION

The `set_lar` command sets attributes in the `HW_CONFIG` configuration object. The attribute to be set can be selected using the `–A` *attr* option (1 = `DEFECT`, 2 = `ERROR`). Attribute 1 `DEFECT` is set if no attribute is specified.

If attribute 1 `DEFECT` is set, the following additional actions are performed:

The `set_lar` command checks whether a command to set the board to `DEFECT` is specified for this board type in the `ctlr_defect.tab` table. If this is the case, this command is executed.

Activities on the board are then stopped insofar as possible. Whether the activities can be stopped and how they are to be stopped is defined using the `ctlr_deact.tab` table. If a deactivation command is configured in this table and it was possible to execute the command correctly, the `DEACTIVE` flag in the `HW_CONFIG` configuration object is set with the appropriate values.

OPTIONS

`–A` *attr*
: Specifies the attribute number (1 or 2). Default: 1.

`–c` *cabinet_ID*
: Specifies the logical cabinet number.

`–b` *bus_id*
: Specifies the bus ID.

`–s` *slot_ID*
: Specifies the slot number of the controller. If this option is not specified, the action is performed for the entire IO box.

`–n` *channel*
: Specifies the channel number on the controller.

`–i` *id*
: ID to be entered in the configuration object. Default: 0.

`–e` *err*
: Error key to be entered in the configuration object. Default: 0.

`–m` *comment*
: Comment to be entered in the configuration object.

  Default: id = 0:
  : "deactivated by operator" or "set by operator"

  Default: id = 1:
  : ""

  Default: id = 2:
  : "deactivated by LAR" or "set by LAR"

DIAGNOSTICS

Return values have the following meaning:

`1 =` Invalid parameter or option

`2 =` An error during editing of `HW_CONFIG`

NOTES

Components that were put out of operation by LAR functions are no longer available to the user after the next reboot.

If a slot is not operational, its channels also cannot be accessed.

FILES

`/usr/bin/lar/ctlr_deact.tab`
`/usr/bin/lar/ctlr_defect.tab`
`/etc/init.d/lar_rc`
`/var/lar/hwconfig`

SEE ALSO

`act_dev`(1M), `deact_dev`(1M), `reset_lar`(1M), `lar`(7), `olr`(7).

System Administrator's Reference Manual

setcolor(1M)
RM400 only

NAME

`setcolor, setcolour -` set screen color

SYNOPSIS

`setcolor` [`–nbo`] *color* [*color*]

`setcolour` [`–nbo`] *color* [*color*]

DESCRIPTION

`setcolor` and `setcolour` allow the user to set the screen to a specific color. Both foreground and background colors can be set independently in a range of 16 colors. `setcolor` with no arguments produces a usage message that displays all available colors, then resets the screen to its previous state.

For example, the following strings are possible colors:

`blue`	magenta	brown	black
`lt_blue`	lt_magenta	yellow	gray
`cyan`	white	green	red
`lt_cyan`	lt_white	lt_green	lt_red

OPTIONS

In the arguments below, *color* is taken from the above list.

`–n` Sets the screen to normal white characters on black background.

color [*color*]
Sets the foreground to the first color. Sets background to second color if a second color choice is specified.

`–b` *color*
Sets the background to the specified color.

`–o` *color*
Sets the color of the screen border (over scan region). To reset border color, use `–o black`.

NOTES

The ability of `setcolor` to set any of these described functions is ultimately dependent on the ability of devices to support them. For example, the `–o` option does not work on the Color Graphics Adapter (CGA).

`setcolor` emits an escape sequence that may or may not have an effect on monochrome devices.

Occasionally changing the screen color can help prolong the life of your monitor.

XENIX `setcolor` command options (`–r`, `–g`, `–p`, `–c`) are ignored.

System Administrator's Reference Manual

sethdlcpc(1M)

NAME

`sethdlcpc -` initialize a TCP/IP link via HDLC-Inhouse

SYNOPSIS

`sethdlcpc` [`debug`] `start` | `stop` *dev-name host-ip-name* [*linklabel*]

DESCRIPTION

For every PC that wants to communicate with UNIX servers via TCP/IP-HDLC, a separate protocol stack must be created and an IP routing entry (`ifconfig`) must be made in the system routing table. The `sethdlcpc` initialization program is responsible for setting up this stack and making routing entries.

A HDLC link is generally activated directly after installation with the configuration tool `termadd`. The connection is cleared using `termdel` and the configuration is deleted.

PARAMETERS

`debug`

Additional information is output that would otherwise only be given in the log file `/var/spool/locks/sethdlcpc.`*tty-name*.

`start`

Activates the HDLC link.

`stop` Deactivates the HDLC link.

dev-name

Path and name for the TTY node.

host-ip-name

The IP name specified here is the subnetwork address of the HOST in the newly configured HDLC subnetwork. The name must be defined in `/etc/hosts`.

linklabel

A linklabel is also possible and `hdlcpc` is the default label. The linklabel is sought in `/etc/hdlcdefs` and the link-specific parameters entered after this label are transferred and set for this link in the HDLC driver. If no linklabel is possible, the defaults for the HDLC driver are used. Only the following HDLC parameters can be changed at present:

`blocked | unblocked`

blocked or unblocked mode

`framelength`

Max. HDLC frame length (in bytes)

`iactpoll`

Active poll cycle for HDLC link (in ms)

EXAMPLES

```
sethdlcpc   start   /dev/term/hdlcp1  ciscoxy
```

In this example, the HDLC link "hdlcp1" is activated with the default values for the HDLC line parameters. The PC "hdlcp1" can then establish a connection to the host "ciscoxy" via TCP/IP.

```
sethdlcpc   stop   /dev/term/hdlcp1
```

This command deactivates the HDLC link that was activated above. In general, a HDLC link is deactivated using the configuration tool `termdel` before the deinstallation.

SEE ALSO

`termadm`(1M), `termdel`(1M).

System Administrator's Reference Manual

setikdb(1M)

NAME

`setikdb -` enable/disable the internal kernel debugger

SYNOPSIS

`setikdb [ on | off ]`

DESCRIPTION

`setikdb on` enables the ikdb.

`setikdb off` disables the ikdb.

`setikdb` without arguments shows the actual status.

DIAGNOSTICS

On successful completion, `setikdb` returns 0, otherwise 1 is returned.

NOTES

Only the superuser may use `setikdb`.

System Administrator's Reference Manual

setmnt(1M)

NAME

`setmnt` - establish mount table

SYNOPSIS

`/sbin/setmnt`

DESCRIPTION

`setmnt` creates the `/etc/mnttab` table which is needed for both the `mount` and `umount` commands. `setmnt` reads standard input and creates a `mnttab` entry for each line. Input lines have the format:

filesys node

where *filesys* is the name of the file system's "special file" (such as `/dev/ios0/sdisk*s?`) and *node* is the root name of that file system. Thus *filesys* and *node* become the first two strings in the mount table entry.

NOTES

Problems may occur if *filesys* or *node* are longer than 32 characters. `setmnt` silently enforces an upper limit on the maximum number of `mnttab` entries.

FILES

`/etc/mnttab`

SEE ALSO

`mount`(1M), `mnttab`(4).

System Administrator's Reference Manual

setuname(1M)

NAME

`setuname` - change machine information

SYNOPSIS

`setuname` {`–s` *name* | `–n` *node*} [`–t`]

DESCRIPTION

`setuname` changes the parameter value for the system name and node name. Each parameter can be changed using `setuname` and the appropriate option.

OPTIONS

`–s` Changes the system name. *name* specifies new system name and can consist of alphanumeric characters and the special characters dash, underscore, and dollar sign. The first character must be an alpha character.

Note:

If the system name is changed, a new installation or update installation of the system cannot be carried out successfully.

`–n` Changes the node name. *node* specifies the new network node name and can consist of alphanumeric characters and the special characters dash, underscore, and dollar sign.

`–t` Temporary change. No attempt will be made to create a permanent change.

Either or both the `–s` and `–n` options must be given when invoking `setuname`.

The system architecture may place requirements on the size of the system and network node name. The command will issue a fatal warning message and an error message if the name entered is incompatible with the system requirements.

LOCALE

The `LC_MESSAGES` environment variable governs the language in which message texts are displayed.

The `LC_CTYPE` environment variable governs character classes, character conversion (shifting) and the behavior of character classes in regular expressions.

If `LC_MESSAGES` or `LC_CTYPE` is undefined or is defined as the null string, it defaults to the value of `LANG`. If `LANG` is likewise undefined or null, the system acts as if it were not internationalized.

If any of the locale variables has an invalid value, the system acts as if none of the variables was set.

The `LC_ALL` environment variable governs the entire locale. `LC_ALL` takes precedence over all the other environment variables that affect internationalization.

NOTES

`setuname` attempts to change the parameter values in two places: the running kernel and, as necessary per implementation, to cross system reboots. A temporary change only affects the running kernel.

System Administrator's Reference Manual

setup(1M)

NAME

`setup` - initialize system for first user

SYNOPSIS

`setup`

DESCRIPTION

The `setup` command, which is also accessible as a login by the same name, allows the first user to be established as the "owner" of the machine.

The user can set the date, time, and time zone of the machine.

The user can set the system language and international environment.

The user can set the node name of the machine.

The user can protect the system from unauthorized modification of the machine configuration and software by giving passwords to the administrative and maintenance functions. Normally, the first user of the machine enters this command through the setup login, which initially has no password, and then gives passwords to the various functions in the system. Any that the user leaves without password protection can be exercised by anyone.

The user can then give passwords to system logins such as "bin" (*provided they do not already have passwords*). Once given a password, each login can only be changed by that login or "root".

Finally, the user is permitted to add the first logins to the system, usually starting with his or her own.

The user can define default values for the memory dumps.

DIAGNOSTICS

The `passwd(1)` command complains if the password provided does not meet its standards.

NOTES

If the setup login is not under password control, anyone can put passwords on the other functions.

SEE ALSO

`passwd(1)` .

System Administrator's Reference Manual

shared_dev(1M)
RM600 only

NAME

`shared_dev` - handling SCSI devices with multiple CPU access (shareddevice)

SYNOPSIS

`shared_dev [–a][–b][–u][–h]`

DESCRIPTION

Devices are accessed from a variety of hosts in multihosted SCSI environments (RMS, Oracle). The serial number is used to identify the SCSI devices in the case of multiple access (disks, tapes, CD-ROMs, printers).

The `shared_dev –a` or `–b` command uses the assistance of the `clos_sh`(1M) command to read `.daps` files (Data Pool Service) on the accessible nodes and then generates the devices with multiple CPU access for the relevant node(s) based on the serial number.

The `shared_dev –u` command identifies the devices with multiple CPU access, reads the system time and configuration data on the accessible nodes and uses the `clos_sh`(1M) and `clos_cmd`(1M) commands to generate the current (latest) error flags for the identified devices.

OPTIONS

No options specified:
The `–b` option is valid.

`–a` Establishes which devices have multiple system access on behalf of the distributed administration system.

`–b` Establishes which devices have multiple system access on behalf of the local nodes.

`–u` Establishes which devices have multiple system access on behalf of the local nodes and distributes the current error flags for devices with multiple CPU access in the distributed administration system.

`–h` Usage for this command.

DIAGNOSTICS

The return values have the following meaning:

`1` Error processing `shared_dev`.

`2` Invalid parameter or option.

NOTES

The command currently only evaluates SCSI disks with serial numbers.

The `SIclos` package [see `clos`(7)] must be installed.

SEE ALSO

`act_dev`(1M), `deact_dev`(1M), `reset_lar`(1M), `clos`(7), `lar`(7), `olr`(7).

System Administrator's Reference Manual

shmcntl(1M)

NAME

`shmcntl` - control shared memory policy and activate shared memoryalignment

SYNOPSIS

`shmcntl` [`-a` {`-l`|`-g`}] *command*

DESCRIPTION

The `shmcntl` command controls how shared memory is allocated and aligned to 64-Mbyte memory addresses and then executes the *command* command for the specified allocation conditions in the required alignment mode. On an RM600-E, a board can request memory be allocated globally (`round robin`, distributed equally among boards) or locally. If `shmcntl` is called without options or explicitly with the `-a` option, the shared memory is aligned to a 64-Mbyte address.

Alignment

Alignment here means that both the start address and the length of the shared memory area can be divided evenly by 64 Mb.

Aligning the shared memory is very important for reducing memory usage in the operating system, as page translation tables for all processes that use the shared memory area must only be created once for the shared memory required. The address translation tables can only be created in a physical memory area of between 0 and 512 Mbytes in the case of operating system versions prior to V5.44; when this limit is reached, further increases in system performance cannot be expected.

Without alignment, the memory used for these tables is proportional to the product of the shared memory size and the number of user processes. In large configurations, this can easily run to hundreds of Mbytes. This type of critical utilization of the lower memory area is effectively avoided by aligning the shared memory areas.

Alignment can be done in two ways:

1) Explicitly by parameterizing `shmat` calls within applications [see `shmat(2)` in `shmop(2)`]. This procedure must be used directly when creating a shared memory application.

2) Automatically by calling applications under the control of `shmcntl`. The advantage of this procedure is subsequent applicability without needing to intervene in existing applications.

Two aspects must however be noted:

a) The `shmat(2)` system call does produce automatic alignment, but does not comply with the X/OPEN standard.

b) The virtual address space is increased in affected processes with the following characteristics:

- The virtual memory address is aligned automatically to 64 Mb if an address is preset to zero or to an address divisible by 64 Mb when the shared memory area is being allocated. The alignment mode is passed on within the process group of the specified *command* command via `fork(2)` and `exec(2)` .
- The size of the shared memory area is extended simultaneously to the next highest multiple of 64 Mb. This can result in a considerable increase in the virtual process size. An increase in the "ulimits" may therefore be necessary for affected processes. The swap area may also need to be increased across the whole system.
- If there are a number of neighboring shared memory areas, alignment of the start addresses is shifted to the next memory areas.
- All accesses are legal within shared memory areas, even accesses on memory areas that are allocated implicitly. For that reason, no application address errors are returned.

OPTIONS

No option specified:
Shared memory is aligned as described above.

`-a` Same as if no option is specified.

`-l` (RM600-E only) The shared memory is requested locally by a board on which the *command* command is running.

`-g` (RM600-E only) The shared memory is distributed equally (`round robin`) across all boards.

EXAMPLES

```
shmcntl quickdd if=file of=/dev/ios0/rstape004
```

In this example, the specified `quickdd` command is executed under the activated control of the 64 Mbyte alignment. An address translation table is only created once for the shared memory areas created in the `quickdd` command, although `quickdd` creates a number of internal processes (read and write processes).

The command is only executed in this mode if the virtual address space was previously set > □128□Mb using `ulimit -v`.

SEE ALSO

`ulimit`(1), `quickdd`(1M), `swap`(1M), `execv`(2), `shmop`(2).

System Administrator's Reference Manual

show_defects(1M)

NAME

show_defects - show LAR entries

SYNOPSIS

show_defects [–t *taglist*] [–w *linelength*] [start | stop | local | repair]

DESCRIPTION

The show_defects command shows LAR entries (deactive, defect, error). If no parameter is specified, the entries are displayed for all nodes in a cluster configuration (CADMIN).

OPTIONS

–t Shows the entries for the nodes specified on the *taglist*. A list comprises the tag names separated by commas. A "white space character" (blank, tabulator, newline character) terminates the list.

–w The line length of the screen output is modified in accordance with *linelength*. Values less than 62 are ignored; the default is 80.

start
Call from rc2, only display locally with old display format.

stop Call from rc0, no activities.

local
Only display locally, not cluster-wide; a show_lar(1M) command is called, if it exists, i.□e. the information is displayed in the new show_lar format.

repair
Only display locally, not cluster-wide, with old unabbreviated display format, i.□e. additional fields are output.

EXAMPLES

The show_defects start call produces the following output, for example:

```
HW_CONFIG defects table
type  :name         :log:cab:bid:slt:chn:attribute:set by    :cause
SPbus :CP4-V02      :   :0  :0  :2  :1  :error    :fw_check  :fw_check 11
SPbus :HIOS         :6  :0  :0  :8  :0  :defect   :operator  :set by oper
Fan   :(hw address )   :0  :   :1  :   :error    :system    :warning
```

The show_defects repair call produces the following output, for example:

```
HW_CONFIG defects table:
type  :name         :log:cab:bid:slt:chn:serial number   :value:time\
stamp:attribute:set by      :cause
SPbus :EHIOS        :   :0  :0  :10 :   :000000001093    :11\
:917420749 :error :fw_check    :fw_check 11:firmware rev. for module
EHIOS is  01.36, should be 01.39
```

SEE ALSO

lar(7).

System Administrator's Reference Manual

show_lar(1M)
RM600 only

NAME

`show_lar -` show LAR entries

SYNOPSIS

`show_lar [-s] [-u] [-c] [-d] [-F] [-C] [-a] [-e] [-T] [-N] [-E] [-n] [-v] [-L]`

DESCRIPTION

The `show_lar` command shows LAR entries (`deactive`, `defect`, `error`), whereby the higher-level hierarchies are also displayed to get a better impression of the location of the controller. If no option is specified, the following fields are displayed by default: `Name`, `level`, `attribute`, `set by` and `cause`. The `Name` and `level` fields are indented hierarchically.

OPTIONS

No option specified:
Corresponds to the specification of the `-c`, `-d` and `-F` options.

`-s` Shows system entries.

`-u` Shows UPS entries.

`-c` Shows cabinet entries.

`-d` Shows SCSI device entries.

`-F` Shows fiber channel device entries (FC).
Extended output (the `cause` field is omitted here).

`-C` Shows configuration data.

`-a` Shows HW-CONFIG address.

`-e` Shows error number.

`-T` Shows time stamp.

`-N` Shows serial number.

`-E` Shows whether memory module is entered in the EEPROM.

`-n` Shows whether an NVRAM entry exists.

`-v` Verbose: Shows all controllers, even those without an HW-CONFIG entry.

`-L` Long output: Output of the `cause` and `configuration data` fields is not shortened.

EXAMPLES

Invoking `show_lar` without options produces the following output for example:

```
HW_CONFIG cabinet table
-----------------------
Name              | level     |attribute|set by  |cause                      |
SYSTEM_T5    SY64|cab :    0|         |        |                           |
   SP-BUS-CAGE    | busid:000|         |        |                           |
      CP4-V02     |  slot:  2|         |        |                           |
         cpu   1  |   chan: 1|error    |fw_check|fw_check 11:firmware rev.  |
      HIOS        |  slot:  8|         |        |                           |
         scsi  6  |   chan: 0|defect   |operator|set by operator            |
+  FAN CAGE       | cage:   7|         |        |                           |
+     fan unit 0  |  slot:  1|error    |system  |warning                    |
```

SEE ALSO

lar(7).

System Administrator's Reference Manual

shutdown(1M-ucb)
BSD Compatibility Package

NAME

`shutdown` - close down the system at a given time

SYNOPSIS

`/usr/ucb/shutdown` [`-fhknr`] [*time* [*warning-message* ...]

DESCRIPTION

`shutdown` provides an automated procedure to notify users when the system is to be shut down. *time* specifies when `shutdown` will bring the system down; it may be the word `now` (indicating an immediate shutdown), or it may specify a future time in one of two formats: `+`*number* and *hour*`:`*min*. The first form brings the system down in *number* minutes, and the second brings the system down at the time of day indicated in 24-hour notation.

At intervals that get closer as the apocalypse approaches, warning messages are displayed at terminals of all logged-in users, and of users who have remote mounts on that machine. Five minutes before shutdown, or immediately if shutdown is in less than 5 minutes, logins are disabled by creating `/etc/nologin` and writing a message there. If this file exists when a user attempts to log in, `login(1)` prints its contents and exits. The file is removed just before `shutdown` exits.

At shutdown time a message is written to the system log daemon, `syslogd(1M)`, containing the time of shutdown, the instigator of the shutdown, and the reason. Then a terminate signal is sent to `init(1M)`, which brings the system down to singleuser mode.

The time of the shutdown and the warning message are placed in `/etc/nologin`, which should be used to inform the users as to when the system will be back up, and why it is going down (or anything else).

OPTIONS

As an alternative to the above procedure, these options can be specified:

`-f` Arranges, in the manner of `fastboot`, that when the system is rebooted, the file systems will not be checked.

`-h` Executes `halt(1M)`.

`-k` Simulates system shutdown. Do not actually shut down the system.

`-n` Prevents the normal `sync(1M)` before stopping.

`-r` Executes `reboot(1M)`.

NOTE

Only allows you to bring the system down between `now` and 23:59 if you use the absolute time for shutdown.

FILES

`/etc/nologin`
tells `login(1)` not to let anyone log in

`/etc/xtab`
list of remote hosts that have mounted this host

SEE ALSO

`login(1)`, `fastboot(1M)`, `halt(1M)`, `reboot(1M)`, `sync(1M)`, `syslogd(1M)`, `sync(2)`.

System Administrator's Reference Manual

shutdown(1M)

NAME

`shutdown` - shut down system, change system state

SYNOPSIS

`/usr/sbin/shutdown` [`-y`] [`-g`*grace_period*] [`-i`*init_state*]

DESCRIPTION

This command is executed by the superuser to change the state of the machine. In most cases, it is used to change from the multi-user state (state `2`) to another state (see below).

By default, it brings the system to a state where only the console has access to the Reliant UNIX system. This state is called single-user (see below).

The command sends a warning message and a final message before it starts actual shutdown activities. By default, the command asks for confirmation before it starts shutting down daemons and killing processes.

OPTIONS

`-y` Pre-answers the confirmation question so the command can be run without user intervention. A default of 60 seconds is allowed between the warning message and the final message. Another 60 seconds is allowed between the final message and the confirmation.

`-g`*grace_period*
Allows the superuser to change the number of seconds from the 60-second default.

`-i`*init_state*
Specifies the state that `init(1M)` is to be put in following the warnings, if any. By default, system state "`s`" is used.

Other recommended system state definitions are:

State `0`
Shuts down the machine so it is safe to remove the power. Have the machine remove power if it can. The `rc0` procedure is called to do this work.

State `1`
State `1` is referred to as the administrative state. In state `1` filesystems required for multi-user operations are mounted, and logins requiring access to multi-user filesystems can be used. When the system first comes up into state `1`, only the console is active and other multi-user (state `2`) services are unavailable. Note that not all user processes are stopped when transitioning from multi-user state to state `1`. This mode is used for administrative tasks such as installing optional utility packages.

State `s`, `S`
State `s` (or `S`) is referred to as the single-user state. When the system comes down to single-state, all mounted local file systems remain mounted, and all processes started by `init` that should only be running in multi-user are killed. Additionally, any process that has a `utmp` entry will be killed. This last condition ensures that port monitors and services started by them are killed. The system can only be accessed through the console.

State `5`
Stops the Reliant UNIX system and goes to the SASH.

State `6`
Stops the Reliant UNIX system and reboots to the state defined by the `initdefault` entry in `/etc/inittab`; configure a new bootable operating system, if necessary, before the reboot. The `rc6` procedure is called to do this work.

SEE ALSO

`init`(1M), `rc0`(1M), `rc2`(1M), `rc6`(1M), `inittab`(4).

System Administrator's Reference Manual

smrsh(1M)

NAME

`smrsh` - restricted shell for sendmail

SYNOPSIS

`smrsh –c` *command*

DESCRIPTION

The `smrsh` program is intended as a replacement for `sh` for use in the "Local and Program Mailer" specification in `sendmail`(1M) configuration files. It sharply limits the commands that can be run using the "`|`*program*" syntax of `sendmail` in order to improve the over all security of your system. Briefly, even if a "bad guy" can get sendmail to run a program without going through an alias or forward file, `smrsh` limits the set of programs that he or she can execute.

Briefly, `smrsh` limits programs to be in the directory `/usr/adm/sm.bin`, allowing the system administrator to choose the set of acceptable commands. It also rejects any commands with the characters "`", "<", ">", "|", ";", "&", "$", "(", ")", "\r" (carriage return), or "\n" (newline) on the command line to prevent "end run" attacks.

Initial pathnames on programs are stripped, so forwarding to `/usr/ucb/vacation`, `/usr/bin/vacation`, `/home/server/mydir/bin/vacation`, and `vacation` all actually forward to `/usr/adm/sm.bin/vacation`.

System administrators should be conservative about populating `/usr/adm/sm.bin`. Reasonable additions are `notify`(1), `vacation`(1), and the like. No matter how brow-beaten you may be, never include any shell or shell-like program (such as `perl`) in the `sm.bin` directory. Note that this does not restrict the use of shell or perl scripts in the `sm.bin` directory (using the "`#!`" syntax); it simply disallows execution of arbitrary programs.

COMPILATION

Compilation should be trivial on most systems. You may need to use `–DPATH=\"`*path*`\"` to adjust the default search path (defaults to `/bin:/usr/bin:/usr/ucb`) and/or `–DCMDBIN=\"`*dir*`\"` to change the default program directory (defaults to `/usr/adm/sm.bin`).

FILES

`/usr/adm/sm.bin`
: directory for restricted programs

SEE ALSO

`sendmail`(1M).

System Administrator's Reference Manual

smtp(1M)

NAME

`smtp -` send SMTP mail to a remote host using Simple MailTransfer Protocol

SYNOPSIS

`/usr/lib/mail/surrcmd/smtp` [`–D`] [`–u`] [`–a` *toaddr*] [`–d` *domain*] [`–H` *helohost*] *sender host recip ...*

DESCRIPTION

`smtp` sends a message to a remote host *host* using the Simple Mail Transfer Protocol (SMTP). The message is read from standard input. *sender* is used to identify the sender of the message and the *recip*s are used as the recipients.

When establishing a connection, `smtp` will use the first transport for which `netdir_getbyname` [see `netdir`(3N)] returns an address, based on hostname, transport [returned from `getnetpath`(3N)], and service `smtp`. Normally, this will be the "tcp" transport.

OPTIONS

`–D` This option turns on debugging. Debugging information is printed to stderr.

`–H` *helohost*
: This option can be used to set the hostname used in `SMTP HELO` message (this defaults to the system's name).

`–d` *domain*
: This option can be used to set the domain name to be used for this host.

`–u` Assumes it is unix format and does no conversion of destination address to RFC822 format.

`–a` *toaddr*
: Opens connection to *toaddr* instead of *host* specified.

`smtp` is normally run by the `smtpsched` process to deliver mail queued in `/var/spool/smtpq`.

FILES

`/var/spool/smtpq`
: where messages are queued

SEE ALSO

`smtpsched`(1M) , `getnetpath`(3N) , `netdir`(3N) .

RFC821 – Simple Mail Transfer Protocol.

System Administrator's Reference Manual

smtpd(1M)

NAME

`smtpd` - receive incoming SMTP messages

SYNOPSIS

`/usr/lib/mail/surrcmd/smtpd` [`-n`] [`-H` *helohost*] [`-h` *thishost*] [`-L` *loadlim*] [`-l` *maxprocs*]

DESCRIPTION

`smtpd` is a daemon that normally runs while in multiuser mode, waiting for requests from remote hosts to send mail. `smtpd` listens for these requests on any TLI-based network for which the SMTP service is defined to [see `netdir_getbyname` in `netdir`(3N)]. As requests are received, `smtpd` will fork off child `smtpd` processes to handle each individual SMTP transaction.

OPTIONS

`-n` Do not create `smtpsched` processes to process the incoming mail. Rely on the hourly `cron`(1M) invocation of `smtpsched` instead.

`-H` *helohost*
: This option can be used to specify the name to be used for the host in the initial `SMTP HELO` message. If it is not specified, the name used in the `HELO` message defaults to the system node name.

`-h` *thishost*
: Specifies the network name to be prepended to the sender path in the `From` line of the message.

`-L` *loadlim*
: Specifies the maximum load at which `smtpd` will create children. If this option is not specified, there is no limit to the load at which children may run. The load is determined by reading the kernel variable `avenrun`.

`-l` *maxprocs*
: This option is used to specify the maximum number of children of `smtpd` that can be running at once. Each child handles one SMTP conversation. If this option is not specified, there is no limit to the number of children that may run.

Mail that is successfully received is put in the appropriate subdirectory under `/var/spool/smtpq` and `smtpsched` is started, if the `-n` option is not given. A log of all `smtpd`'s activities is kept in the file `/var/spool/smtpq/LOG`.

FILES

`/dev/kmem`
: to get the current machine load (avenrun)

`/etc/services`
: list of TCP/UDP services (SMTP should be 25/tcp)

`/etc/net/*/services`
: list of other TLI networks' services

`/usr/lib/mail/surrcmd/fromsmtp`
: where incoming mail is piped to

`/var/spool/smtpq/LOG`
: log of `smtpd` transactions

SEE ALSO

`cron`(1M) , `fromsmtp`(1M) , `smtp`(1M) , `netdir`(3N) .

System Administrator's Reference Manual

smtpqer(1M)

NAME

`smtpqer` - queue mail for delivery by SMTP

SYNOPSIS

`/usr/lib/mail/surrcmd/smtpqer` [`–nu`] [`–a` *toaddr*] [`–d` *domain*] [`–H` *helohost*][`–N`] [`–D`] *sender host recip ...*

DESCRIPTION

`smtpqer` queues the mail message it reads from standard input for eventual delivery by `smtp`. The message is queued for delivery to the host specified in the *to* address.

`smtpqer` should normally be invoked by the `mail` command by placing the following line in `/etc/mail/mailsurr`:

```
'.+' '([^!@]+)!(.+)' '< /usr/lib/mail/surrcmd/smtpqer %R \\1 \\2'
```

`smtpqer` will check the host name in the *to* address. If it is one that can be reached (i.e. if `netdir_getbyname` [see `netdir`(3N)] can find it on at least one TLI network), the message will be queued, and `smtpqer` will exit with a return code of 0 (which means the mail was successfully queued). Otherwise, it will return with an exit code of 1, and the message will not be queued.

Messages that are queued are stored in a file under the SMTP queue directory (`/var/spool/smtpq`).

The `–H` option is used to specify the host name that should be used in the `SMTP HELO` message. This option is passed to the `smtp` program.

By default, the `–N` option is set in the file `/etc/mail/mailsurr` and means that MX resource records (MX = Mail Exchanger) are not taken into account. MX information is provided by the DNS (Domain Name Service). An MX resource record specifies a mail distributor, i.e. a system that can forward messages to a domain or to specific computers in a domain.

The `–D` option is used to write detailed information about mail routing to `/var/spool/smtpq/log`.

The `–d` option is used to specify the domain name that should be used for your host. This option is passed to the `smtp` program. If this option is not used, and a domain has been specified in the mail configuration file `mailcnfg`, that domain will be used instead.

The `–u` option prevents the mail's headers from being converted to RFC822 format. This option is passed to the `smtp`(1M) program.

The `–a` option is used to specify the "to address" that is passed to the `smtp` program. Finally, the `–n` option is used to prevent `smtpqer` from starting an `smtpsched` process to deliver the mail.

FILES

`/usr/bin/rmail`
: where mail originates from

`/etc/hosts`
: database of remote hosts (for TCP/IP)

`/etc/mail/mailcnfg`
: mail configuration file

`/etc/net/*/hosts`
: database of remote hosts (for other TLI networks)

`/etc/mail/mailsurr`
: control file containing rule to invoke `smtpqer`

`/usr/lib/mail/surrcmd/smtpsched`
: program to process message queues

`/usr/lib/mail/surrcmd/smtp`

program that passes message to remote host

`/usr/lib/mail/surrcmd/tosmtp`
filter to convert to RFC822 format

`/var/spool/smtpq`
where messages are queued

SEE ALSO

`rmail`(1), `smtpsched`(1M), `smtp`(1M), `netdir`(3N).

RFC822 – Standard for the Format of ARPA Internet Text Messages.

System Administrator's Reference Manual

smtpsched(1M)

NAME

`smtpsched` - process messages queued in the SMTP mail queue

SYNOPSIS

`/usr/lib/mail/surrcmd/smtpsched` [`-c`] [`-d`] [`-v`] [`-t`] [`-s` *scheds*] [`-r` *days*] [`-w` *days*] [*qnames*]

DESCRIPTION

`smtpsched` is used to process the messages queued in the SMTP mail queue `/var/spool/smtpq`. It is invoked automatically by the SMTP mail surrogate `smtpqer`, whenever mail is queued for SMTP delivery to a remote host, and by `smtpd` whenever incoming mail arrives. It should also be run once per hour (from `cron`) to attempt delivery of any mail that cannot be delivered immediately.

`smtpsched` will normally attempt to send all messages queued under all subdirectories of `/var/spool/smtpq`. However, if *qnames* are specified, only those listed subdirectories of `/var/spool/smtpq` will be searched for messages to deliver. The subdirectories each refer to a different remote host.

OPTIONS

`-c` Causes empty queue directories to be removed.

`-v` Causes verbose logging to occur.

`-t` Test mode. The actions `smtpsched` would take are logged but not performed.

`-s` *scheds*
: Specifies the maximum number of concurrent `smtpsched`s that may be running at once. If more than this number is running, `smtpsched` will exit.

`-r` *days*
: Causes mail older than *days* days to be returned.

`-w` *days*
: Any mail older than *days* days will trigger a warning message, which is sent to the originator.

`-d` Debug, gives more log output.

FILES

`/usr/lib/mail/surrcmd/smtp`
: delivers the mail

`/usr/lib/mail/surrcmd/smtpqer`
: queues the mail

`/var/spool/smtpq`
: queued mail messages

`/var/spool/smtpq/LOG*`
: log files

`/var/spool/smtpq/`*host*
: mail messages queued for *host*

SEE ALSO

`cron`(1M) , `smtp`(1M) , `smtpqer`(1M) .

System Administrator's Reference Manual

srtydefs(1M)

NAME

`srtydefs -` manage port-specific settings for the RTY monitor

SYNOPSIS

`srtydefs –a –d` *device_connect_mode* [*options*] *rtylabel*

`srtydefs –l –d` *device_connect_mode* [*rtylabel ...*]

`srtydefs –r –d` *device_connect_mode rtylabel* [*rtylabel ...*]

DESCRIPTION

`srtydefs` manages port-specific settings for the RTY monitor, whereby entries are either made or deleted in the `/var/sadm/terms/rtysettings.d`, `/var/sadm/terms/rtysettings.o` or `/var/sadm/terms/rtysettings.i` file, depending on the *device_connect_mode* (`d`, `i`, `o`). The *rtylabel* labels defined using `srtydefs` are used to configure RTY connections with `SIterms` and `SIconf`. These labels establish the `rtyadm` call executed during configuration.

`srtydefs` may only be invoked by the user `root`.

OPTIONS

`–l –d` *device_connect_mode* [*rtylabel*]
: If *rtylabel* is specified, `srtydefs` shows the data record from the `rtysettings.`*device_connect_mode* file, whose label corresponds to the specified *rtylabel*. If *rtylabel* is not specified, `srtydefs` shows the entire contents of the `rtysettings.`*device_connect_mode* file. Output is sent to `stdout`.

`–r –d` *connect_mode rtylabel*
: Deletes every data record with the label *rtylabel* from the `rtysettings` file specified with the `–d` option.

`–a –d` *connect_mode* [*options*] *rtylabel*
: Enters a data record in the `rtysettings.`*device_connect_mode* file, where *rtylabel* is the label used. The content of the data record is defined by the `–f`, `–h`, `–i` and `–s` options. These are only permitted in conjunction with the `–a` option.

The following *options* are possible:

`–f` *flags*
: Specifies one or more flags which define the special features of the RTY connections. The following options are permitted:

  `k` (`KeepAlive`), `f` (`ON_THE_FLY`), `p` (`POLL`) and `e` (`TELNET`).

`–h` *hup_mode*
: Defines the type of host monitoring for `rtymon` connections. The permitted values are: `no_hup`, `imm_hup`, `late_hup`.

`–i` *init_string*
: String sent following establishment of the connection to the LAN device. The characters must be decimal-coded and must be separated by blanks.

`–s` *service*
: Specifies a program to be called before the `rtymux` driver is linked to the network connection. The specified program must exist and must be executable.

DIAGNOSTICS

If `srtydefs` is successful, it has the return code `0`. If an error occurs, a value other than zero is returned and an error message is sent to `stderr`.

`1` Configuration error (see `stderr`).

2 Syntactic error.

EXAMPLES

The following command lists all entries for *device_connect_mode* `i`.

```
srtydefs –l –di
```

The following command creates a new data record with the label `mylabel` for *device_connect_mode* `d`. Host monitoring should be of type `late_hup`; the connection should be monitored regularly (`KeepAlive`) and the initstring "27 91 50 121" used.

```
srtydefs –a –dd –hlate_hup –fk –i "27 91 50 121" mylabel
```

The following command deletes all entries with the label `oldlabel` for *device_connect_mode* `o`.

```
srtydefs –r –do oldlabel
```

FILES

```
/var/sadm/terms/rtysettings.i
/var/sadm/terms/rtysettings.d
/var/sadm/terms/rtysettings.o
```

SEE ALSO

`rtyadm`(1M), `rtymon`(1M), `termadd`(1M), `rtty`(7).

System Administrator's Reference Manual

ssync(1M)
RM600 only

NAME

`ssync` - flush system buffers

SYNOPSIS

`ssync` [–v] [–h] [*regular-file* | *filesystem*]

DESCRIPTION

`ssync` causes modified system buffers to be written to hard disk or floppy. `ssync` can be used for individual files, entire file systems, and all active file systems. In contrast to `sync`(1M) , `ssync` is only returned when all affected system buffers have been stored on hard disk or floppy.

OPTIONS

No option specified
: If neither a regular file nor a file system is specified, `ssync` refers to all active file systems.

–v
: Displays all actions when writing to hard disk.

–h
: Displays the syntax for `ssync`.

regular-file
: All modified data and attributes of the specified regular file *regular-file* are written from the system buffer to hard disk.

filesystem
: All modified data and attributes of the specified file system *filesystem* are written from the system buffer to hard disk.

A file system can be specified in 2 ways:

- as the pathname of the directory under which the file system is mounted,
- as the pathname of the device on which the file system is stored.

SEE ALSO

`sync`(1M) .

System Administrator's Reference Manual

strace(1M)

NAME

`strace` - print STREAMS trace messages

SYNOPSIS

`strace` [*mid sid level*] ...

DESCRIPTION

`strace` without arguments writes all STREAMS event trace messages from all drivers and modules to its standard output. These messages come from the STREAMS log driver [see `log(7)`]. If arguments are provided they must be in triplets of the form *mid*, *sid*, *level*, where *mid* is a STREAMS module ID number, *sid* is a sub-ID number, and *level* is a tracing priority level. Each triplet indicates that tracing messages are to be received from the given module/driver, sub-ID (usually indicating minor device), and priority level equal to or less than the given level. The token `all` may be used for any member to indicate no restriction for that attribute.

The format of each trace message output is:

seq time ticks level flags mid sid text

seq trace sequence number

time time of message in *hh*:*mm*:*ss*

ticks
time of message in machine ticks since boot

level
tracing priority level

flags

- `E`: message is also in the error log
- `F`: indicates a fatal error
- `N`: mail was sent to the system administrator

mid module ID number of source

sid sub-ID number of source

text formatted text of the trace message

Once initiated, `strace` will continue to execute until terminated by the user.

EXAMPLES

Output all trace messages from the module or driver whose module ID is 41:

```
strace  41 all all
```

Output those trace messages from driver/module ID 41 with sub-IDs 0, 1, or 2:

```
strace  41 0 1  41 1 1  41 2 0
```

Messages from sub-IDs 0 and 1 must have a tracing level less than or equal to 1. Those from sub-ID 2 must have a tracing level of 0.

NOTES

For performance reasons, only one `strace` process is permitted to open the STREAMS log driver at a time. The log driver has a list of the triplets specified in the command call and compares each potential trace message against this list to decide if it should be formatted and sent up to the `strace` process. Hence, long lists of triplets will have a greater impact on overall STREAMS performance. Running `strace` will have the most impact on the timing of the modules and drivers generating the trace messages that are sent to the

`strace` process. If trace messages are generated faster than the `strace` process can handle them, then some of the messages will be lost. This last case can be determined by examining the sequence numbers on the trace messages output.

SEE ALSO

`log`(7) .

Programmer's Guide: STREAMS

System Administrator's Reference Manual

strclean(1M)

NAME

`strclean -` STREAMS error logger cleanup program

SYNOPSIS

`strclean` [`–d` *logdir*] [`–a` *age*]

DESCRIPTION

`strclean` is used to clean up the STREAMS error logger directory on a regular basis (for example, by using `cron`). By default, all files with names matching `error.*` in `/var/adm/streams` that have not been modified in the last three days are removed. A directory other than `/var/adm/streams` can be specified using the `–d` option. The maximum age in days for a log file can be changed using the `–a` option.

EXAMPLE

```
strclean –d /var/adm/streams –a 3
```

has the same result as running strclean with no arguments.

NOTE

`strclean` is typically run from `cron` on a daily or weekly basis.

FILES

`/var/adm/streams/error.*`

SEE ALSO

`cron`(1M) , `strerr`(1M) .

Programmer's Guide: STREAMS

System Administrator's Reference Manual

strerr(1M)

NAME

`strerr` - STREAMS error logger daemon

SYNOPSIS

`strerr`

DESCRIPTION

`strerr` receives error log messages from the STREAMS log driver [see `log`(7)] and appends them to a log file. The error log files produced reside in the directory `/var/adm/streams`, and are named `error.`*mm-dd*, where *mm* is the month and *dd* is the day of the messages contained in each log file.

The format of an error log message is:

seq time ticks flags mid sid text

seq error sequence number

time time of message in *hh*:*mm*:*ss*

ticks
time of message in machine ticks since boot priority level

flags

- `T`: the message was also sent to a tracing process
- `F`: indicates a fatal error
- `N`: send mail to the system administrator

mid module ID number of source

sid sub-ID number of source

text formatted text of the error message

Messages that appear in the error log are intended to report exceptional conditions that require the attention of the system administrator. Those messages which indicate the total failure of a STREAMS driver or module should have the `F` flag set. Those messages requiring the immediate attention of the administrator will have the `N` flag set, which causes the error logger to send the message to the system administrator via `mail`. The priority level usually has no meaning in the error log but will have meaning if the message is also sent to a tracer process.

Once initiated, `strerr` continues to execute until terminated by the user. It is commonly executed asynchronously.

NOTES

Only one `strerr` process at a time is permitted to open the STREAMS log driver.

If a module or driver is generating a large number of error messages, running the error logger will cause a degradation in STREAMS performance. If a large burst of messages are generated in a short time, the log driver may not be able to deliver some of the messages. This situation is indicated by gaps in the sequence numbering of the messages in the log files.

FILES

`/var/adm/streams/error.`*mm-dd*

SEE ALSO

`log`(7) .

Programmer's Guide: STREAMS

System Administrator's Reference Manual

sttydefs(1M)

NAME

`sttydefs` - maintain line settings and hunt sequences for TTYports

SYNOPSIS

`/usr/sbin/sttydefs –a` *ttylabel* [`–b`] [`–n` *nextlabel*] [`–i` *initial-flags*] [`–f` *final-flags*]

`/usr/sbin/sttydefs –l` [*ttylabel*]

`/usr/sbin/sttydefs –r` *ttylabel*

DESCRIPTION

`sttydefs` is an administrative command that maintains the line settings and hunt sequences for the system's TTY ports by making entries in and deleting entries from the `/etc/ttydefs` file.

`sttydefs` with a `–a` or `–r` option may be invoked only by a privileged user. `sttydefs` with `–l` may be invoked by any user on the system.

OPTIONS

`–l` If a *ttylabel* is specified, `sttydefs` will display the record from `/etc/ttydefs` whose TTY label matches the specified *ttylabel*. If no *ttylabel* is specified, `sttydefs` will display the entire contents of `/etc/ttydefs`. `sttydefs` will verify that each entry it displays is correct and that the entry's *nextlabel* field references an existing *ttylabel*.

`–a` *ttylabel*
: Adds a record to the `ttydefs` file, using *ttylabel* as its label. The following describes the effect of the `–b`, `–n`, `–i`, or `–f` options when used in conjunction with the `–a` option.

`–b` Specifies that autobaud should be enabled. Autobaud allows the system to set the line speed of a given TTY port to the line speed of the device connected to the port without the user's intervention.

`–n` *nextlabel*
: Specifies the value to be used in the *nextlabel* field in `/etc/ttydefs`. If this option is not specified, `sttydefs` will set *nextlabel* equal to *ttylabel*.

`–i` *initial-flags*
: Specifies the value to be used in the *initial-flags* field in `/etc/ttydefs`. *initial-flags* must be in a format recognized by the `stty` command. These flags are used by `ttymon` when searching for the correct baud rate. They are set prior to writing the prompt.

  If this option is not specified, `sttydefs` will set *initial-flags* equal to the `termio(7)` flag `9600`.

`–f` *final-flags*
: Specifies the value to be used in the *final-flags* field in `/etc/ttydefs`. *final-flags* must be in a format recognized by the `stty` command. *final-flags* are the `termio(7)` settings used by `ttymon` after receiving a successful connection request and immediately before invoking the service on the port. If this option is not specified, `sttydefs` will set *final-flags* equal to the `termio(7)` flags `9600` and `sane`.

`–r` *ttylabel*
: Removes any record in the `ttydefs` file that has *ttylabel* as its label.

DIAGNOSTICS

If successful, `sttydefs` will exit with a status of `0`. `sttydefs –l` will generate the requested information and send it to the standard output.

EXAMPLES

The following command will list all the entries in the `ttydefs` file and print an error message for each invalid

entry that is detected.

```
sttydefs –l
```

The following shows a command that requests information for a single label and its output:

```
# sttydefs -l 9600
----------------------------------------------------------------------
9600:9600 hupcl erase ^h:9600 sane ixany tab3 hupcl erase ^h::4800
----------------------------------------------------------------------
ttylabel:       9600
initial flags:  9600 hupcl erase ^h
final flags:    9600 sane ixany tab3 hupcl erase ^h
autobaud:       no
nextlabel:      4800
```

The following sequence of commands will add the labels `1200`, `2400`, `4800`, and `9600` and put them in a circular list:

```
sttydefs –a 1200 –n 2400 –i 1200 –f "1200 sane"
sttydefs –a 2400 –n 4800 –i 2400 –f "2400 sane"
sttydefs –a 4800 –n 9600 –i 4800 –f "4800 sane"
sttydefs –a 9600 –n 1200 –i 9600 –f "9600 sane"
```

FILES

`/etc/ttydefs`

SEE ALSO

`termio(7)`.

System Administrator's Reference Manual

subj(1M)

NAME

`subj` - generate a list of subjects from a document

SYNOPSIS

`subj` *file* ...

DESCRIPTION

`subj` searches *file*(*s*) for subjects that might be appropriate in a subject-page index and prints the list of subjects to standard output. The document should contain formatting commands (from `nroff`, `troff`, and `mm` among others) to make the best use of `subj`.

WARNINGS

`subj` selects sequences of capitalized words as subjects except the first word in each sentence. Thus, if a sentence begins with a proper noun, the capitalization rule will not select this word as a subject. On the other hand, since each sentence is expected to begin on a newline, the first word of a sentence that begins in the middle of a line may be erroneously selected.

The output of `subj` may not be appropriate for your needs and should be edited accordingly.

NOTES

`subj` also selects as subjects modifier-noun sequences from the abstract, headings, and topic sentences (the first sentence in each paragraph), and occasionally a word is incorrectly categorized as a noun or adjective.

SEE ALSO

`mm`(1M), `mmt`(1M), `ndx`(1M), `nroff`(1M), `troff`(1M).

System Administrator's Reference Manual

sulogin(1M)

NAME

`sulogin` - access singleuser mode

SYNOPSIS

`sulogin`

DESCRIPTION

`sulogin` is automatically invoked by `init` when the system is first started. It prompts the user to type the root password to enter system maintenance mode (singleuser mode) or to type EOF (typically <CTRL-D>) for normal startup (multiuser mode). `sulogin` should never be invoked directly by the user.

SEE ALSO

`init`(1M) .

System Administrator's Reference Manual

swap(1M)

NAME

`swap` - swap administrative interface

SYNOPSIS

`/usr/sbin/swap –a` *swapname swaplow swaplen*

`/usr/sbin/swap –d` *swapname swaplow*

`/usr/sbin/swap –l`

`/usr/sbin/swap –s`

DESCRIPTION

`swap` provides a method of adding, deleting, and monitoring the system swap areas used by the memory manager.

OPTIONS

`–a` Adds the specified swap area. *swapname* is the name of the block special partition, e.g. `/dev/ios0/sdisk000s1` or a regular file. *swaplow* is the offset in 512-byte blocks into the partition where the swap area should begin. *swaplen* is the length of the swap area in 512-byte blocks. This option can only be used by the superuser. If additional swap areas are added, it is normally done during the system start up routine `/etc/rc` when going into multiuser mode.

`–d` Deletes the specified swap area. *swapname* is the name of block special partition, e.g. `/dev/ios0/sdisk000s1` or a regular file. *swaplow* is the offset in 512-byte blocks into the the swap area to be deleted. Using this option marks the swap area as "INDEL" (in the process of being deleted). The system will not allocate any new blocks from the area, and will try to free swap blocks from it. The area will remain in use until all blocks from it are freed. This option can be used only by the superuser.

`–l` Lists the status of all the swap areas. The output has five columns:

`path` The path name for the swap area.

`dev` The major/minor device number in decimal if it is a block special device; zeros otherwise.

`swaplo`
The *swaplow* value for the area in 512-byte blocks.

`blocks`
The *swaplen* value for the area in 512-byte blocks.

`free` The number of free 512-byte blocks in the area. This includes the reserved blocks. If the swap area is being deleted, the word INDEL will be printed to the right of this number.

`–s` Prints the following information about total swap space usage:

`allocated`
The amount of swap space (in 512-byte blocks) allocated to private pages.

`reserved`
The number of swap space (in 512-bytes blocks) not currently allocated, but claimed by memory mappings that have not yet created private pages.

`used` The total amount of swap space, in 512-byte blocks, that is either allocated or reserved.

`available`
The total swap space, in 512-byte blocks, that is currently available for future reservation and allocation.

NOTES

To prevent parallel running instances of swap from corrupting internal structures, swap tries to create the lockfile `/tmp/.swapcntl`. If this file exists, another instance of swap is already running, and swap fails. If swap fails although no other swap is running, the lockfile must be removed.

WARNINGS

No check is done to see if a swap area being added overlaps with an existing file system.

System Administrator's Reference Manual

sync(1M)

NAME

`sync` - flush system buffers

SYNOPSIS

```
sync
```

DESCRIPTION

`sync` causes all system buffers not yet written to hard disk or floppies to be flushed out to disk, thus assuring that all file updates up to that point are saved.

A system buffer is an internal buffer that serves to improve system performance. It is not accessible to users.

Application usage

If the system is to be shut down, `sync` should be called beforehand to ensure file system integrity. The system administrator commands `/etc/shutdown` and `/etc/halt` call `sync` automatically.

`sync` will only write local buffers to local disks; you cannot use `sync` to force buffers to be written out to disk on remote machines.

SEE ALSO

`ssync`(1M) , `sync`(2) .

System Administrator's Reference Manual

sysadm(1M)

NAME

`sysadm` - visual interface to perform system administration

SYNOPSIS

`sysadm` [*menu* | *task* | {*menu/task*}]

DESCRIPTION

This command, when invoked without an argument, presents a set of menus that help you do administrative work. If you specify a menu or task on the command line, one of two things happens: if the requested menu or task is unique, it is displayed immediately; if the menu or task is not unique, a menu of choices is displayed.

EXAMPLES

`sysadm users`

(calls up the *users* menu) □
`sysadm make`

(calls the function *Create A File System (make)* in the *filesystems* menu) □
`sysadm users/add`

(calls the function *add* in the *users* menu)

DIAGNOSTICS

The `sysadm` command exits with one of the following values:

`0` = Normal exit.

`2` = Invalid command syntax. Usage message of the `sysadm` command is displayed.

`4` = The menu or task name given as an argument does not exist.

`5` = The menu name given as an argument is an empty placeholder menu, and therefore not available for use.

`7` = The `sysadm` command is not available because it cannot invoke `fmli(1)`. (The FMLI package may be corrupt or it may not have been installed.)

LOCALE

The `LC_MESSAGES` environment variable governs the language in which message texts are displayed.

If `LC_MESSAGES` is undefined or is defined as the null string, it defaults to the value of `LANG`. If `LANG` is likewise undefined or null, the system acts as if it were not internationalized.

If any of the local variables has an invalid value, the system acts as if none of the variables was set.

The `LC_ALL` environment variable governs the entire locale. `LC_ALL` takes precedence over all the other environment variables that affect internationalization.

SEE ALSO

`fmli(1)`.

Manual *System Administration and Hardware Configuration Using the SYSADM User Interface*.

System Administrator's Reference Manual

sysdef(1M)

NAME

`sysdef -` output system definition

SYNOPSIS

`/usr/sbin/sysdef [–n` *namelist* `]`

`/usr/sbin/sysdef –i`

DESCRIPTION

`sysdef` outputs the current system definition in tabular form. It lists all hardware devices, their local bus addresses, and unit count, as well as pseudo devices, system devices, loadable modules, and the values of selected kernel tunable parameters.

It generates the output by analyzing the named bootable operating system file (*namelist*).

The default system *namelist* is `/stand/unix`.

OPTIONS

`–n` *namelist*
: Specifies a *namelist* other than the default (`/stand/unix`). The *namelist* specified must be a valid bootable operating system.

`–i`
: Allows you read the configuration information from the kernel that is currently in memory (i.e. from `/dev/kmem`) rather than from a file.

DIAGNOSTICS

`internal name list overflow`
: If the master table contains more than an internally specified number of entries for use by `nlist`(3E).

FILES

`/stand/unix`
: default operating system file (file that contains the system namelist)

SEE ALSO

`nlist`(3E).

System Administrator's Reference Manual

syslogd(1M)

NAME

`syslogd -` log system messages

SYNOPSIS

`syslogd [ –d ] [ –f` *configfile* `] [ –m` *interval* `] [ –o ] [ –O` *path* `] [ –P ] [ –l SINIX-Y ] [ –h ]`

DESCRIPTION

`syslogd` reads system messages and application messages [sent via the `syslog(3C)` programming interface] from the STREAMS `/dev/log` multiplexer.

`/dev/osm2` is also read if `syslogd` has been called with the `–o` option.

The messages read by `/dev/log` contain flags which indicate the "priority" of the message, detail the "level" of importance of the message and the "facility" of the message. The values are specified as parameters when the message is sent via the `syslog(3C)` programming interface. The possible values are defined with 8 "levels" and 24 "facilities" in `/usr/include/sys/syslog.h` [see `syslog(5)`].

The priority is used by `syslogd` to decide the destinations to which the messages are to be forwarded. The messages (or message text) are forwarded to their destinations without the priority but are preceded by a time stamp which has been generated by `syslogd`. *Note*: The format of the messages output is not described in any greater detail here since it may not always be the same.

The destinations recognized by `syslogd` include pipes, files, users, other hosts, and Logging V3.0 (= LOG3.0).

The number of destinations is limited to 20.

Pipes should only be used if it can be guaranteed that the messages will always be read at the receiving end, otherwise any further logging may be blocked.

Files and pipes are not created as destinations, rather must simply exist. Destinations that can neither be reached nor written are ignored. Messages can be sent simultaneously to several destinations (multiplexing).

The `/etc/syslog.conf` configuration file is used to define the allocation of messages with a specific priority to particular destinations [see `syslog.conf(4)`]. The syntax must be adhered to strictly here. In particular, it must be noted that priorities ("selector") and allocated destinations ("actions") have to be separated by at least one tab.

If `syslogd` has been called with the `–m` *interval* option, the daemon generates a time message every *interval* minutes (= sign of life) with the message text "`-- MARK --`". The default for the time message is 0 minutes (= "no time message"). The time messages can likewise be forwarded to particular destinations using the configuration file. The time messages are identified by the term `mark`.

`syslogd` reads the configuration file when it starts up, and again whenever it receives a HUP signal, at which time it also closes all files it has open, re-reads its configuration file, and then opens only the log files that are listed in that file. `syslogd` exits when it receives a `TERM` signal.

As it starts up, `syslogd` creates the `/etc/syslog.pid` file containing its process ID (PID), if possible.

OPTIONS

`–d` Turns on debugging.

`–f` *configfile*
: Specifies an alternate configuration file.

`–m` *interval*
: Specifies an interval, in minutes, between mark messages.

`–o` Reads and logs `/dev/osm2`.

`–O` *path*

Changes the path of the device to be read (e.g. from the default `/dev/osm2` to `/dev/osm`); this is only effective if the `-o` option is also specified.

`-P` Forwards the priority in plain text to the destinations (e.g. `user.info`); this is useful for testing purposes.

`-l SINIX-Y`
(only valid for RM600): Logs the contents of NVRAM. This option is useful so that after a panic-fall and restart, the last messages before the crash can be made available.

`-h` Displays usage.

FILES

`/etc/syslog.conf`
Configuration file

`/etc/syslog.pid`
Process ID

`/dev/log`
STREAMS log driver

`/dev/osm2`
Special device

SEE ALSO

`logger(1)`, `syslog(3C)`, `syslog.conf(4)`, `syslog(5)`, `log(7)`.

tbl(1M) - tunestat(1M)

tbl(1M)

NAME

`tbl` - prepare tables for nroff or troff

SYNOPSIS

`tbl` [`-TX`] [`--`] [*file* ...] [`-`]

DESCRIPTION

`tbl` is a preprocessor that prepares tables for `nroff(1M)` or `troff(1M)`. `tbl` assumes that lines between the `.TS` and `.TE` command lines describe tables, thus they are re-formatted; lines outside these command lines are copied to the standard output. (`tbl` does not alter the `.TS` and `.TE` command lines.)

The `-TX` option forces `tbl` to use only full vertical line motions, making the output more suitable for devices that cannot generate partial vertical line motions (for example, line printers).

If no you give no file names as arguments, or if you specify `-` as *file*, `tbl` reads the standard input, so it may be used as a filter. When you use it with `eqn(1M)` or `neqn(1M)`, put `tbl` first to minimize the volume of data passed through pipes.

Macros

`.TS` Table start.

`.TS H`
Table start with header definition which appears on each continuation page of the table.

`.TH` Defines the end of the table header.

`.TE` Table end.

`.T&` A *new* format within a table is initiated with `.T&` (see *Examples*).

Options

Follow `.TS` with global options. These options are globally valid for the complete table and must be closed with a semicolon (`;`). The available global options are (lower or uppercase letters):

`center`
: Centers the table (default is left-adjust);

Note: The British spelling "centre" is not allowed.

`expand`
: Makes the table as wide as the current line length;

`box` Encloses the table in a box;

`doublebox`
: Encloses the table in a double box;

`allbox`
: Encloses each item of the table in a box;

`tab`(*x*)
: Uses the character *x* instead of a tab to separate items in a line of input data;

`linesize`(*n*)
: Sets line or rules (e.g. from `box`) in point size *n*.

Format descriptions

After global options come lines describing the format of each line of the table. Each such format line describes one line of the table itself, except that the last format line (which you must end with a period) describes *all* remaining lines of the table. A single key letter describes each column of each line of the table. You may follow this key letter with specifiers that determine the font and point size of the corresponding item, that indicate where vertical bars are to appear between columns, and that determine column width, inter-column spacing, etc. The available key-letters are (lower or uppercase letters):

`c` Centers item within the column.

`r` Right-adjusts item within the column.

`l` Left-adjusts item within the column.

`n` Numerically adjusts item in the column. Numeric column entries are justified as follows (see *Examples*):

- by the decimal point located at the furthest right position to a digit;
- by the furthest digit to the right-hand side;
- by the optionally insertable control character `\&`

`s` Horizontally spanned column(s): entry of previous column is continued (see *Examples*).

`a` Alphabetical sub-column. The longest entry is centered, all further ones are left-justified after this one (see *Example 2*).

`^` Vertically spanned line(s): entry of upper line is continued and vertically centered (see *Examples*).

`^t` Vertically spanned line(s): entry of upper line is continued and started in the top line (see *Example 4*).

`_` Replaces this entry with a horizontal line.

`=` Replaces this entry with a double horizontal line.

Possible additional specifications:

`B`, `I` Set font for the column (`B` = bold, `I` = italic).

f*n*, f*c*, f*cc*
: Set font for the column: Values for *n* (digit) or *c* or *cc* (1 or 2 characters) correspond to values set up via the request `.fp` *n* (see *Examples*).

`e` All thus defined columns are formatted to equal width (see *Example 2*).

`u` Entries are superscripted ½ line (see *Example 2*).

`z` Length of entry is ignored at calculation of column width, nevertheless, the entry is printed in whole

length (see *Example 7*).

`w(n)` *Minimum* width of column (see *Examples*).

`p`*n* Determination of the point size (analogous to `.ps`) (see *Example 2* and *6*).

`v`*n* Determination of the line height (analogous to `.vs`). Is of effect only in text blocks marked `T{`␣...␣`T}` (see *Example 2*).

n A number *n* as additional specification determines the distance between the columns in `n`-units (default is `3n`) (see *Example 3* and *5*).

`|`, `||` Vertical single or double line between the columns.

Table entries

The format lines are followed by lines containing the actual data (table entries) for the table, followed finally by `.TE`. Within such data lines, data items are normally separated by tab characters.

Table entries consist of specifications for the columns defined in the format description. Possible entries are (cf. *Example 2*):

text Text specification, may be empty.

`T{` Text block which may be several lines long

text (see *Examples*).

`T}`

`_`, `=`
Single or double line across the whole column width (adjacent to neighboring columns).

As individual entry: single or double line across the complete table width (see *Example 2*).

`\_` Single line across the width of the entry (not adjacent to neighboring columns) (see *Example 2*).

`\R`*c* Repeats a character *c* across the width of the entry (not adjacent to neighboring columns) (see *Example 2*).

`\^` Performs only in this line the format description of the above located entry of the previous line; equivalent to the format description `^`.

`\a` Draws a dotted line to the end of a column (always use together with the width specification `w(n)` of a column (see *Example 2* and *6*).

Note:

All table entries – except for simple text specifications – must not contain *any blanks* before or after the column separation character.

No more than *maximally 30 text blocks* may be used in a table.

EXAMPLES

Example 1:

If we let *T represent a tab (which should be typed as a genuine tab), then the input:

```
 .TS␣
center box;␣
cB s s␣
cI | cI s␣
^  | c  c␣
l  | n  n.␣
Household Population␣
_␣
Town\*THouseholds␣
\*TNumber\*TSize␣
=␣
Bedminster\*T789\*T3.26␣
```

```
Bernards Twp.\*T3087\*T3.74□
Bernardsville\*T2018\*T3.30□
Bound Brook\*T3425\*T3.04□
Bridgewater\*T7897\*T3.81□
 .TE
```

yields (the output is different depending on `nroff` or `troff` formatting): □

□□□□□□Household Population		
Town	□□□□Households	
	Number	Size
Bedminster	789	3.26
Bernards Twp.	3087	3.74
Bernardsville	2018	3.30
Bound Brook	3425	3.04
Bridgewater	7897	3.81

Example 2:

The input

```
 .TS□
tab(#), center, box;□
cf3p+2 s s s.□
Spanned, centered main heading□
_□
 .T&□
l | cfI s s□
^ | l || r || c.□
"vertically centered" # centered italic heading□
#_#_#_□
 .sp .5□
# leftttt # right # centered□
# left # rightttttt # ce□
# left # rightt # centered□
=□
 .T&□
lw(30n) | ne | lue | rw(1c)e□
aw(30n) | ne | le | rw(1c)ev+5.□
 .sp .5□
left-justified main column# 0.05 # lefttt-up # right□
alphab. sub-column# 26.04.87 # left # rightttt□
short# 43\&57.2 ## right□
# 0.675#\_#\Rx□
long□
text# 874 #_#\R.□
short#00.56□
---- # ---- #=#T{□
wrapped-around
long text block;□
text□
T}□
 .TE
```

yields:

Spanned, centered main heading

"vertically centered"	centered italic heading		
	leftttt left left	right righttttt rightt	centered ce centered
left-justified main column	0.05	lefttt-up	right
alphab. sub-column	26.04.87	left	righttttt
short	43 57.2		right
	0.675	_____ __	xxxxxxxx
long text	874	_____ __	.....
short	00.56		
----	----		wrapped-around long text block; text

Example 3: Table with `s` ("span")

The input

```
 .TS
tab(!), box, center;
cf2 s |cf2 s
l2 l |l2 l.
UNIX ! MS-DOS
_
ASCII ! ASCII ! Pixel ! Pixel
Mouse ! ! ! Scanner
! ! Color
 .TE
```

yields:

UNIX		MS-DOS	
ASCII	ASCII	Pixel	Pixel
Mouse			Scanner
		Color	

Example 4: Table with vertically spanned lines at top (`^t`)

The input

```
 .TS
tab(#), center;
lf3| l| lf3.
deciduous ## coniferous
_
 .T&
l| l| l
l| ^t| l.
 .sp .5
```

```
oak #T{□
\f2Central European□
 .br□
trees\f1□
T}# spruce□
elm ## larch□
birch ## pine□
beech ## silver fir□
poplar ## juniper□
chestnut ## yew□
 .TE
```

yields:

deciduous		coniferous
oak	*Central European* □ *trees*	spruce
elm		larch
birch		pine
beech		silver fir
poplar		juniper
chestnut		yew

Example 5: Table with automatic entry counting

The input

```
 .TS□
tab(#), center;□
l1 l n1 l.□
 .nr X 0 1□
 \n+X) # Chandelier, Crystal, 12 Candles # 5500 # Kcs□
 \n+X) # Candlestick, Light Balls with Chains # 1800 # Kcs□
 \n+X) # Table Clock, Wooden # 700 # Kcs□
 .nr Y 0 1□
 .af Y a□
    \n+Y) # Tureen□
    \n+Y) # Oval Bowl□
    \n+Y) # Oval Bowl with a Handle□
    \n+Y) # Bowl, Triangular Form□
 \n+X) # Detto Amphora # 300 # Kcs□
 \n+X) # Detto Drinking Service # 1000 # Kcs□
 .nr□
YY
0 1□
 .af YY a□
    \n+(YY) # 1 Can□
    \n+(YY) # Cup with a Handle□
 \n+X) # Vase, Hand-ground # 400 # Kcs□
 \n+X) # Drinking-cups, 6 Pieces, Metal Edge # 240 # Kcs□
 \n+X) # Drinking-glasses, 4 Pieces, with Golfers # 120 # Kcs□
 .TE
```

yields:

1) Chandelier, Crystal, 12 Candles 5500 Kcs

2)	Candlestick, Light Balls with Chains	1800	Kcs
3)	Table Clock, Wooden	700	Kcs
□□ a)	Tureen		
□□b)	Oval Bowl		
□□c)	Oval Bowl with a Handle		
□□d)	Bowl, Triangular Form		
4)	Detto Amphora	300	Kcs
5)	Detto Drinking Service	1000	Kcs
□□a)	1 Can		
□□b)	Cup with a Handle		
6)	Vase, Hand-ground	400	Kcs
7)	Drinking-cups, 6 Pieces, Metal Edge	240	Kcs
8)	Drinking-glasses, 4 Pieces, with Golfers	120	Kcs

A table with "automatic" counting of table entries is created when the `nroff`/`troff` requests `.nr` and `.af` are used. A *numeric* register is set when the `.nr` request is used. The exact syntax of a numeric register is: `.nr` *Name N M* where *N* specifies the start value and *M* the increment/decrement value. In our example, the register is named "X", the start value is 0 and the increment value is 1. The register "X" is incremented by the value 1 with each interpolation (copying of the register value) according to the syntax: `\n+`*Name*.

A *Format* is assigned to a register *Name* when the request `.af` *Name Format* is used. This means in our example: first the "Y" is set, as described above. Then format "a" is assigned to register "Y" (= alphabetic counting with lowercase Latin letters). Finally, register "Y" is again incremented with each interpolation.

Attention: The registers must be set *within* the table.

Example 6: Table with dotted lines (`\a`)

The input

```
 .TS□
tab(#), center;□
lw(30n) cBp10 s s□
lw(30n) c s s□
lw(30n) cBp10 cBp10 cBp10□
lw(30n) cBp10 s s□
lw(30n) n n n.□
#Annual Gross Balance, December 31st□
#_□
#1988#1987#1986□
#(Dollars in Millions)□
 .sp□
Premiums\a#$\010,922.7#$\010,330.7#$\0\09,252.4□
Investments\a#3,671.7#3,146.0#2,749.7□
Income Taxes\a#24.4#91.6#71.9□
Net Incomes\a#375.2#373.1#312.4□
Assets\a#41,645.8#34,434.7#32,876.6□
  .TE
```

yields:

Annual Gross Balance, December 31st		
1988	1987	1986

□□□□□□□□□□□□□□(Dollars in Millions)

Premiums.....	$ 10,922.7	$ 10,330.7	$ 9,252.4
Investments.....	3,671.7	3,146.0	2,749.7
Income Taxes.....	24.4	91.6	71.9
Net Incomes.....	375.2	373.1	312.4
Assets.....	41,645.8	34,434.7	32,876.6

A dotted line after a table entry is achieved by entering `\a`. It is important to enter the length of the column with `w(n)` and the entry of `\a` at the same time. If this is not performed, the column setup will be destroyed (in our example: `30n` in the format descriptions).

Note: The entries `\0` are not part of the `tbl` syntax, but normal `nroff`/`troff` special functions which are used to create a blank character with the width of the digit zero. They are only for aesthetic reasons to ensure a correct spacing with the $ characters in the columns.

Example 7: Table with overlapping columns (`z`)

The format description `z` causes the length of an entry to ignored when the column width is calculated. The function can now be used to generate overlapping columns in place of parallel columns, as it is normally the case. The distribution of entries over various columns leads to juxtaposition of entries, as can be seen in this example. The input

```
 .TS□
tab(!);□
lw(5n)z lw(5n)z lw(10n)z l l1 lw(2n)z1 lw(5n)z.□
!!!!Space□
 .sp□
!!Dimensionality!!!!Non-Dimensionality□
 .sp□
!Horizontality!!Verticality!!Surface□
 .sp□
Prospectivity!!Laterality!!!Volume□
 .TE
```

yields:

		Space		
	Dimensionality			Non-Dimensionality
	Horizontality	Verticality		Surface
Prospectivity	Laterality			Volume

NOTES

There are still some problems when using `tbl` to construct multi-page boxed tables with continuation headings (`.TS□H`, `.TH`). In some cases, boxes are not properly constructed (distorted pages, loops) after crossing a page boundary and the table drops into the page feed trap. In this situation, the macro `.ch #f` must be entered at the point at which the page feed occurs in the output (it disables the trap mechanism).

In addition, with multi-page tables with boxes or vertical lines, the macro `.T#` should be inserted at the point at which the page feed is to take place.

SEE ALSO

`nroff`(1M), `troff`(1M).

REFERENCE

M. E. Lesk, *TBL — A Program To Format Tables*

D. Dougherty, T. O'Reilly, *UNIX TEXT PROCESSING*, Hayden Books, Indianapolis, Indiana 46268 USA, ISBN 0-672-46291-5

System Administrator's Reference Manual

tc(1M)

NAME

`tc` - troff output interpreter

SYNOPSIS

`tc` [`–t`] [`–o` *list*] [`–a`□`n`] [`–e`] [*file*]

DESCRIPTION

`tc` interprets its input (standard input default) as output from `troff`(1M) . The standard output of `tc` is intended for a TEKTRONIX 4015 (a 4014 terminal with ASCII and APL character sets). The various typesetter sizes are mapped into the 4014's four sizes; the entire `troff` character set is drawn using the 4014's character generator, using overstruck combinations where necessary, producing an altogether displeasing effect. Typical usage:

```
troff file | tc
```

At the end of each page `tc` waits for a new-line (empty line) from the keyboard before continuing on to the next page. In this wait state, the following commands are recognized:

`!`*cmd* Send *cmd* to the shell.

`e` Invert state of the screen erase

–*n* Skip backward *n* pages.

a*n* Set the aspect ratio to *n*.

`?` Print list of available options.

OPTIONS

`–t` Does not wait between pages (for directing output into a file).

`–o` *list*
Prints only the pages enumerated in *list*. The list consists of pages and page ranges (e.g. 5–17) separated by commas. The range *n*– goes from *n* to the end; the range –*n* goes from the beginning to and including page *n*.

`–a` *n* Sets the aspect ratio to *n*; default is 1.5.

`–e` Does not erase before each page.

NOTES

Font distinctions are lost.

It needs a `–w` option to wait for input to arrive.

SEE ALSO

`nroff`(1M) , `troff`(1M) .

System Administrator's Reference Manual

termadd(1M)

NAME

`termadd -` configure device

SYNOPSIS

`termadd –?` □
`termadd –T?` □
`termadd –b?` □
`termadd –T? –r?` □
`termadd –T? –b` *boardname* □
`termadd –b` *boardname*`?` □
`termadd –T` *type* \–s{*nationality*|?} \ [–k{*keyboard*|?}] □
`termadd –T` *type* \–k{*keyboard*|?} \ [–s{*nationality*|?}] □
`termadd –F?` □
`termadd –F` *config-file* □
`termadd –T` *type* `–r` \ *hostname*,*portno* [*add_options*] *name* □
`termadd –T` *type* `–b` \ *boardname*,*portno* [*add_options*] *name* □
`termadd –T` *type* `–c` \ *hostterm* [*add_options*] *name* □
`termadd !`

BASIC EXAMPLES

`termadd –T tc10-vt –r annex,0 –u tc10`
A device of type `tc10-vt` is connected to the TACLAN annex, connection 0. The name reads "tc10". This connection can also be used by UUCP.

`termadd –T generic –b chase0,1 –p "ID: " nonsni`
Anon-SNI terminal is connected to the chase card 0, port 1; "ID: " is used as the prompt.

`termadd –Tserial –b itp0,0 –n waage1`
A serial device is connected to the itp card, port 0. The device is called "waage1".

`termadd –T sinix-te –r pc,0,PC –o etype=97801-5,channel=0+1 pc`
Two 97801-5 emulations are configured on a PC named "pc".

DESCRIPTION

The `termadd` command performs the complete configuration of a device of type *type* and name *name*. The relevant entries are created in the file system, the appropriate STREAMS configuration is performed, and the login and possibly the loading service is assigned via SAF (Service Access Facility).

`termadd` can be called in two ways. A single device can be configured using the call arguments, or any number of devices can be configured using a file with a call (`F` option). Configuration of a device is always transaction-oriented, i.e. either the entire configuration process is executed successfully, or a partial configuration which has already been made is completely reversed, regardless of how many jobs (still) need to be processed. This means that in the case of a mass configuration, the procedure continues with the next job. Mass configurations are typically activated by configuration generators such as `Config`. However, configuration via this interface requires that the calling software executes all checks carried out by `termadd` in the case of individual jobs. More particularly, `termadd` no longer executes any syntax checks. The device parameters are specified in a `:` notation. The order is output using the help option (`–F?`). However, the meanings of the individual fields cannot be derived from the options on the individual job interface. In this context, the "mass interface" is not intended for general use, but is reserved for configuration generators.

A special algorithm rules out the possibility of `termadd` running parallel to `termdel`. However, if either the `termadd` or `termdel` command ended due to a system error, it is likely that neither `termadd` nor `termdel` is operable.

All available device connection options on UNIX systems can be configured using `termadd`, provided that

- when connecting to TACLAN or directly to the Ethernet, the `r` option is used (see above)
- when connecting directly to a card or via connection concentrators, the `b` option is used
- when connecting to a device which is already configured (see below), the device name (`c` option) is used.

It is no longer possible to specify the connection using device numbers (major/minor device number).

Handling secondary devices

Some terminals (such as the ba80) allow secondary devices to be connected. These secondary devices are configured by naming the "primary terminal" as a connection (`c` option). This can be canceled by calling "`termadd`
`!`" next. "Primary terminals" are essentially terminals with the name part "ba80" or "–cluster" or "sinix-te". Only devices that do not have a connection for secondary devices by themselves can be configured as secondary devices.

Handling PCs

In principle, PCs and printers can be connected in three different ways:

- directly to a controller; (configured using the `b` option);
- via a TACLAN to the Ethernet; (configured using the `r` option);
- directly to the Ethernet (can also be configured using the `r` option – in this case, the PC is considered to be a TACLAN.

When installing the terminal emulation "sinix-te" on PC, the HOST side must be configured using the terminal type "sinix-te".

OPTIONS

All options and arguments should be specified as described under `intro_sarm1(1M)` . It is particularly important to ensure that the arguments in the options do not contain any blanks [see also `getopts(1)`].

The specifications for *type* and *boardtype* may not contain any wildcards (e.g. `–T ba8*`).

`–?` Outputs help information.

`–T?` Outputs all supported device types.

`–b?` Outputs all controllers permitted for this type of machine.

`–T? –b` *boardname*
Outputs a list of all the device types supported which can be connected to this controller or card.

`–T? –r?`
Outputs a list of how the supported device types are connected to the Ethernet (STREAMS, RTELNET).

`–b` *boardtype?*
Outputs the syntax and an example of a configuration of a board of this type.

The workstation program or the code tables for a terminal depend on both the type of keyboard and on the country-specific characters. Terminal emulation may also be a consideration here. The relationship between the type of terminal, the keyboard and country-specific characters, the workstation program, the `TERM` variables and the line parameters is set up in the `/var/sadm/terms/choices` file. `termadd` checks this file in accordance with the `T`, `s` and `k` options, and determines the relevant settings if an appropriate entry exists, or sets default settings if no suitable entry was found. The country codes are specified by the "ISO 3166 alpha-2 country code". Exceptions arise where several codes are possible for one country (for example, `CH.FR` as a French variant of a Swiss keyboard), or one code applies to several countries (`AT/DE` for the countries of Germany and Austria, for example). Common keyboard type identifiers include `ct..` for ba80 keyboards, and `mf2` or `UNIX-7-bit` for 97801 keyboards.

As the country-specific characters can be extended in any system, a number of additional information functions are available which indicate the current range of functions offered.

The `–T` *type* option outputs the type of device to be configured. This option may not be omitted when

specifying the following options and must always be specified first. Information on the terminal type used is written in the comment field of the relevant SAF entry and cannot be changed.

−T *type* −s?
: Outputs all country-specific characters for the specified terminal type.

−T *type* −k?
: Outputs all the keyboards permitted for this terminal type.

−T *type* −k? −s?
: Outputs all country-specific characters and all keyboard types for the specified terminal type.

−T *type* −s? −k *keyboard*
: Outputs a list specifying all country-specific characters which are available for the specified keyboard of the specified type of terminal.

−T *type* −k? −s *nationality*
: Outputs a list containing all keyboard types which are available for the specified terminal type and the specified country-specific characters.

−T *type* −s *nationality* −k *keyboard*
: Definition of a place program or a code table independent of the settings in the terminal itself. If a *nationality* or a *keyboard* is missing, either the default form or the default keyboard is selected. In this case, the file `/var/sadm/terms/choices` is searched from the beginning until the first suitable line is found.

−T *type* −r *hostname*,*portno*[,*tsvrtype*]
: Specifies an Ethernet connection: *hostname* specifies the name of the terminal server, printer server, PC, or LAN printer, while *portno* specifies the device number via which the device is addressed. A distinction is made between "IN", "OUT" and "DATA" connections (`rty` configuration). In the case of "IN" connections, the connection is established on the initiative of the device connected to a terminal or printer server or directly to the Ethernet. Several different hosts can generally be selected for the device with this type of configuration. With "OUT" and "DATA" connections, on the other hand, the connection is established on the initiative of the host which, for example, starts a login service on the connected device. In the case of such configurations, the connected device is driven by precisely one host. It is not possible to select a host on the device side.

  The device type (*tsvrtype*) can also be specified (default: "annex8"). Any alphanumeric character string is permitted as the *tsvrtype*. A controlling monitor (`ttymon`) is assigned to each TACLAN which manages all connected devices. If a device is connected directly to the network, a monitor is created for each device. This can be avoided by generating the monitor with a common pseudo tsvrtype, for example all LAN printers with "PRT" and all PCs with "LAN".

  The *portno* argument is generally a decimal number. "DATA", "OUT" and "IN" configurations are distinguished by preceding the port number with the letter "d", "o" or "i". The default value is "o" and can be omitted. "DATA" and "IN" configurations are only possible if the device can be operated by means of the STREAMS procedure (RTY). All devices that use the `dptg` protocol are not supported, i.□e. the terminal types ba80, ba80vt100, ba47, 9766, ba80-pc, sinix-te and 9766-pc. In the case of "IN" devices, the *portno* argument defines the local tcp port number on which the `rtymon` listens for incoming connection requests. With "OUT" and "DATA" devices, this argument represents the tcp port number of the remote device. This number is a relative number in the case of the tsvrtypes "annex8", "O1", "O2" "PC" and "R", i.□e. the device basis (7000 for ANNEX device) is included internally (for compatibility reasons). The individual arguments must be separated by a comma (","); blanks are not permitted.

  The terminal server and printer server connections themselves are configured using the `tsvrset`(1M) command.

−T *type* −r *hostname*,*portno*[,*tsvrtype*]
: Specifies an ETHERNET connection: *hostname* specifies the name of the TACLAN, PC, or LAN printer, and the *portno* of the connection number via which the device is addressed. The device type *tsvrtype* can also be specified (default setting: `O2`). The permitted range of values is specified in the

/var/sadm/terms/rtydefs file, where the "hosts" are listed together with the configuration of important attributes (e.g. basic address, see below). The values O1, 02, and R represent the different TACLAN models. The LAN printer is indicated by PRT, and the value PC refers to the PCs that are directly connected to the LAN. Special characters are not permitted.

The *portno* argument must contain a decimal number. In connection with the types O1, 02, R, and PC, the relative port numbers are specified, e.g. the device-related basic address 7000 is calculated internally. For LAN printers, the port number must be specified absolutely (basic address 0). The arguments must be specified by a comma (","), and blanks are not permitted.

Please ensure that the TACLAN connections are configured correctly (see **NOTES**).

A controlling monitor (ttymon) is assigned to each TACLAN. If PC was specified as the type (see above), a single monitor has control over all PCs (name: PCM).

−T *type* −b *boardname*,*portno*

Specifies a direct connection: *boardname* represents the card type and its number, *portno* represents the number of the connection to which the terminal is connected. Depending on the card type, the argument *boardname* can be unstructured. If this is the case, the information function specifies the correct syntax.

The *portno* argument must contain a decimal number. The syntax differs, depending on the board/controller (in particular: sih, sin). The correct structure of the b option is output with termadd −b*boardname*?, together with an example.

The arguments *boardname* and *portno* must be separated by a comma (","). Blanks are not permitted.

TACLAN connections cannot be configured in this way.

−T *type* −c *hostterm*

Some device types can also be connected to a host terminal as a secondary device. Only terminals of the type 97801 cluster, ba80, 9766, or ba47 can have a secondary device (ba80, 9766 and ba47 only on channel 8 or 9). *hostterm* is the name of the host terminal.

−n *name*

Specifies the name of the device. This name must be self-explanatory to the user. Configuration details, such as board number, should no longer appear in the name because all information required for configuration is output using the termshow command. The device name is relative to /dev/term and should contain a maximum of 6 characters. In later versions, UNIX also permits longer names (up to 26 characters). The relevant upper limit is accepted automatically each time SIterms is restored. The name of the device can also be given as an argument. However, if the -n option is also specified, this option has priority.

The following specifications are possible for *add_options*:

−t *pmtag*

Specifies the monitor (ttymon) that provides the login service. This name may contain a maximum of 11 characters. Special characters are not permitted. If no monitor is running with the specified *pmtag*, it is created. The monitor created writes a log file called /var/saf/*pmtag*/log. This file name is entered in the list /var/sadm/terms/cronlist, and cron ensures that these files do not become too large by calling the /usr/bin/lograng command (see crontab −l).

If the −t option is not specified, termadd creates a name for the terminal monitor from the specifications for the b or r option (*boardname* or TACLAN name).

The −t option cannot be specified before the −b option (see above).

−l *linesetting*

Specifies the label *linesetting* from the /etc/ttydefs file which defines the stty arguments.

−m *rtysetting*

Specifies the label *rtysetting* from the /var/sadm/terms/rtysettings.*dev_conn_mode* file which defines the rty arguments, where *dev_conn_mode* is the prefix "d", "i" or "o" of the *portno* from the −r option. The −m option is only permitted in conjunction with the −r option.

`-u` With this specification, a configured connection can also be used by means of UUCP. Terminals with this configuration do not issue the prompt until the CR key has been pressed. This functionality is not available for all device types.

`-p` *prompt*
Specifies the character string for the prompt. Placeholders such as `#TERM` are no longer permitted.

`-d` Deactivation flag: There is no LOGIN available on the specified device.

`-v` *verbose*
Controls the range of diagnostic output. Values for *verbose*:

`0` Only error messages are displayed.

`1` Both error messages and warnings are displayed (default).

`>1` Outputs diagnostic information which may not be relevant for the user.

`-o` *special_option*
Special options can be used for some device types. The `-o` option can only be used as the last option for the `termadd` command.

Arguments for the terminal type ba80/9766/ba47:

`channel=`*number*[`+`*number*`+` ...]
Specifies which terminal channel is to be configured (default: 0). Every terminal of type ba80, 9766, or ba47 must be configured from channel 0. *number* should be in the range 0 to 15. If a number of channels are to be created simultaneously, they are linked using "+". The channel specifications need not be specified already sorted.

`muxid=`*id*
Specifies the STREAMS multiplexer which controls this terminal. Permitted values range between 0 and 9. By default, a regular assignment is defined by a special algorithm. If a value is specified, this overrides the default. This option can only be specified for channel 0.

Arguments for terminal emulations on PCs:

`etype=97801-5|generic|ba80|97801-480`
Specifies the type of emulation.

Arguments for the 97801-5, 97801-480, 97801-cluster, and 97808-32 terminals:

`emulation=7|8`
Specifies the type of emulation for this terminal. Either a 7-bit or 8-bit emulation is possible (default: 8-bit). The terminal software is determined by the program based on the language and the emulation. Information on the emulation used is written in the comment field of the relevant SAF entry.

Arguments for the `hdlc` terminals (name part "hdlc", cash registers, or MWSS devices):

`channel=`*number*[`+`*number*`+` ...]
Specifies which terminal channels are to be configured (default: 0). Each hdlc terminal must be configured as channel 0. *number* should be within the range of values 1 to 15. If a number of channels are being created simultaneously, they are linked using "+". The channel entries need not be specified already sorted.

With `channel=t`, a TCP/IP configuration can be configured using hdlc connections [see `sethdlcpc`(1M)].

The specification `channel=s` is also permitted. This configures operation without the DPTG2 protocol.

`linklabel`
Specifies the label in `/etc/hdlcdefs`, which defines the LINK-related HDLC line parameters.

`linelabel`
Specifies the label in `/etc/hdlcdefs`, which defines the string-related HDLC line parameters.

`router`

This specification is only permissible with a TCP configuration via HDLC, and specifies the name of the host that operates the HDLC route.

`ctras=`*number*[+*number*+ ...]
This argument shows that for these terminals, a "ctras" service must be set up instead of the LOGIN service. The special channels are specified in the same way as for `channel`.

`mwssdev`, `mwssctab`, and `mwssactive`
These arguments are intended for an MWSS application only and are only worth implementing in conjunction with IOCS and MWSS-SW. Further information can be found in the corresponding documentation (DevC MWSS V3.0, Konfigurieren und Verwalten (German only), Order no. U22493-J-Z765-1).

Arguments for the loadable tc20 devices (tc20-ansi, tc20-vt, tc20-97801):

`load=`*number*
number specifies the firmware variant which is loaded, or is to be loaded in the flashprom of the device. Valid variants are to be found in the directory `/etc/tc20`. If `termadd` finds the default version (*number*`.default`) for the specified variant, a `xbload` call is generated which compares the loaded version with the default version, and where necessary, dynamically loads the (newer) `default` version. If *number*`.default` does not exist, then the configuration is rejected. If the tc20 devices are configured without the argument `load`, no `xbload` call is generated.

NOTES

`termadd` is concerned with the STREAMS configuration of terminals of the type ba80/9766 and with the daemon processes of terminals connected to a host system over an Ethernet network. A table is maintained (`/var/sadm/terms/termtab`) by `termadd`, which is read by `termadm` when bringing the system up or down. Network configurations must be carried out using the `tsvrset`(1M) and `tsvrget`(1M) commands.

You can use the `-k undef` and `-s undef` options to prevent a terminal program being loaded for terminals of type 9766/ba80 or 97801/97808. If the workstation program of such terminals is not to be determined from the system but from the terminal itself, the `-k` and `-s` options cannot be used.

The 97801-53 terminal can only be connected via a SIM board to an internal TAK (`itak`), port 0.

`termadd` can only be called by the system administrator (`root`).

`termadd` has been internationalized, i.e. the messages output by `termadd` can be translated into any language in accordance with `setlocale`(3C), `gettxt`(1), and `mkmsgs`(1). The English version is the default and is always used if no value is assigned to the variable `LANG`.

Call the command `termdel` to delete a device configuration.

The `termshow` command outputs all the information required for configuring the device as a `termadd` call; this information may be used directly to reconfigure this device if it is (inadvertently) deleted by `termdel`.

If ba80, 9766, and related terminals, that work with the DPTG2 protocol are connected to the TACLAN, special care must be taken to ensure that the ports on the TACLAN are configured correctly:

need_dsr:	`N`
mode:	`slave`
control_lines:	`both`
type:	`dial_in`
bidirectional_modem:	`N`

These parameters can be set using the terminal server commands `na` or `admin`.

A file of default settings (`/etc/default/SIterms`) is available containing variables used by `termadd`. In this file, you can set the assignments between the terminals and the respective monitor programs by changing the maximum number of terminals that the relevant monitor can manage.

DIAGNOSTICS

All messages are written to the `stderr` file. The value 0 is returned after the call is executed successfully. In the event of an error, a value other than null is returned.

`1` = Problem with the configuration (see `stderr`).

`2` = Syntax error.

When calling via a configuration file (`F` option), a status message is output for each device configured.

EXAMPLES

`termadd –Tba80 –b itp0,2 kat`
A terminal "kat" of type ba80 is connected to itp board 0, port 2.

`termadd –T 97801-5 –b itp0,2 –t mymonitor –l 97801 –sAT/DE –v0 nice`
`termadd` for a terminal of type 97801-5 (e.g. a 97801-524 terminal) with the language `AT/DE` (Austria/Germany). The terminal is connected to the ITP board, second port, and is assigned the name "nice". The terminal monitor "mymonitor" provides the login service. Only error messages are output (`–v 0`). The line parameters are determined via the "97801" flag.

`termadd –T 9766 -b echase0,unit1,n2 –s GB –k ct08 –o muxid=2 beavis`
`termadd` carries out a configuration for channel 0 of a 9766 terminal. The terminal is connected to echase board 0, unit 1, port 2. A modem cannot be located at port 2 (n2). Ports to which modems can be attached are specified without "`n`". The keyboard is of type "ct08", and the language is "GB". The DPTG protocol of the terminal is handled by multiplexer no. 2 (`muxid=2`).

`termadd –T 9766 –b itp0,12 –o channel=8 dummy`
`termadd` is executed for channel 8 of a terminal of type 9766 whose channel 0 must already been configured at port 12 of itp board no. 0. Channel 8 is assigned the name "dummy". `termadd` recognizes the channel 8 function on this terminal (a printer port) and therefore does not create a login service on this channel.

`termadd –Tprinter –c printer –n nd90`
An "nd90" printer is connected to the "printer" port (channel 8, see above) with the `–c` option.

`termadd –Tba80 –b sim0,line2,8tak1,v3`
A terminal is to be set up for TACSI; a TAK with 8 ports is used, and the TAK ports are called "v" for V.24, "n" for non-modem, and "s" for SS97 ports.

`termadd –T97801-53 –b sim0,line2,itak1,s0 L7`
A terminal of type 97801-53 has an internal TAK, connected directly to the SIM board. The terminal can only be operated on the SIM board via `itak`, port `s0`.

`termadd –Tba80 –r annex3,4 rock`
A ba80 terminal "rock" is connected to the TACLAN "annex3"; the device is configured for "slave" mode; "annex8" is accepted as the default TACLAN type. The terminal server or printer server is configured using the `tsvrset(1M)` command.

`termadd –Ttc10-vt –r tsvr,m7411,topos cli`
A TC10-vt terminal "cli" is connected to the terminal server "tsvr"; the device is configured for "master" mode; "topos" is set as the TACLAN type. The terminal server or printer server is configured using the `tsvrset(1M)` command.

```
termadd –Tba80-hdlc –b sih1,SW8i0,222 –lba80-mod \
–ochannel=3,linelabel=L2 hdlc
```
A ba80-hdlc terminal "hdlc" with the device address 222 is connected to sim board 1. `linelabel` (L2) refers to special hdlc line settings for controllers/line 5. The tag ba80-mod indicates an entry in `ttydefs` which is suitable for modem operation (external clocking).

```
termadd –Tserial-hdlc –b sih1,SW8i0,111 –nbeetle \
–ochannel=s,linklabel=pos2000
```
A "beetle" POS terminal with the device address 1 is connected to the sih card 0, string "i0" via a connection box of the type "SW8" with the device address "111". The entry `channel=s` indicates

operation without the DPTG2 protocol. The entry `linklabel` (pos2000) refers to the special hdlc line settings for this connection. A login service is not provided for this connection. The device is not loaded ("serial" type).

`termadd –Tsinix-te –r mypc,0,PC –n mypc –ochannel=1+8+2+3+4+0,ctras=2+4`
APC "mypc" is connected via Ethernet. "PC" is assigned as a monitor. The channels 0, 1, 2, 3, 4, and 8 are created simultaneously; channels 2 and 4 have CTRAS facilities. The terminal emulation is the default (ba80).

`termadd –Tba80-hdlc-pc –b sih1,SW8Rv1,123 –o channel=6 kh`
A hdlc terminal emulation (screen 6) called "kh" is connected to sih board 1, line "v1" at a connection box of type "SW8R" with device address "123". The relevant 0 channel must already be configured. Specifications can be made either in uppercase or lowercase.

`termadd –T97801-5 –b motherboard0,5 –n rom –o emulation=7 rom`
A 97801-5 terminal called "rom" is operated directly at port 5 of the CPU board "motherboard", and a 7-bit terminal emulation is set. The relevant line parameters are set by `termadd`.

`termadd –Tprinter –b slp3,0 –l special bigpr`
A "bigprint" printer is connected to "data products board" 3. The printer is configured in such a way that the connection of a terminal monitor is monitored, which sets the required "special" line parameters every time the printer is switched on. The terminal monitor naturally does not provide a LOGIN.

`termadd –T tc20-ansi –b tca4,5 –s MN scanc`
`termadd –T tc10-ansi –b tca4,5 –s MN scanc`
The ANSI emulation of a TCx0-v100 terminal is configured with this call. If a keyboard variant is to be used, whose respective code table is not "integrated" in the terminal (e.g. the table for Mongolia), the `s` option must be used. In this case, the terminal is switched from `SIterms` to SCANCODE mode. SCANCODE mode is only (implicitly) possible in such cases. Applications that want to work with SCANCODES must configure the terminal for "normal mode" and carry out the switch to SCANCODES or back to normal mode.

LOCALE

The `LC_MESSAGES` environment variable governs the language in which message texts are displayed.

If `LC_MESSAGES` is undefined or is defined as the null string, it defaults to the value of `LANG`. If `LANG` is likewise undefined or null, the system acts as if it were not internationalized.

If any of the locale variables has an invalid value, the system acts as if none of the variables was set.

The `LC_ALL` environment variable governs the entire locale. `LC_ALL` takes precedence over all the other environment variables that affect internationalization.

FILES

```
/dev/term
/etc/ttydefs
/etc/hdlcdefs
/etc/ttytype
/etc/hosts
/etc/termtab
/usr/bin/terms/cronlist
/var/sadm/terms/cronlist
/var/sadm/terms/choices
/var/sadm/terms/hdlclabs
/var/sadm/terms/rtydefs
```

SEE ALSO

`getopts(1)`, `login(1)`, `intro_sarm1(1M)`, `sac(1M)`, `termadm(1M)`, `termdel(1M)`,

termshow(1M), tsvrget(1M), tsvrset(1M), ttymon(1M).

System Administrator's Reference Manual

termadm(1M)

NAME

`termadm -` device administration

SYNOPSIS

`termadm` [–v *verbose*] *tag* [*hostname* [*devname* [*portno* [*devtype*]]]]

`termadm –?`

DESCRIPTION

In the context of device configuration [see `termadd(1M)` or `termdel(1M)`], `termadm` is used for managing devices connected (i.e. serial devices, printers or PCs) via TACLAN or the MUX protocol (terminal type 97801-51xP) or DPTG2 V1.0 (on another host system). `termadd` processes a table (`termtab`), which is created or updated by `termadd` or `termdel`.

This table contains a series of fields separated from one another by a colon (:). The last field contains a command which is always executed when a line is selected. The first five arguments of the command `termadm` correspond to columns in `termtab`. `termadm` searches through this table and selects a line if the argument in the command line corresponds to a value in the relevant column in the table. If two or more arguments are specified, each of these arguments must have a corresponding value in the relevant column in the table; otherwise, this line is not selected. `termadm` executes the command contained in the last field in the table for the selected lines.

OPTIONS

Every argument specifies a value in the `termtab` table; this value must correspond to a column in the `termtab` table so that the line can be selected. The placeholder * matches every value. It must be enclosed in single quotes (').

`–?` Outputs help information.

`–v` *verbose*

Controls the range of diagnostic output. Values for *verbose*:

`0` Only error messages are displayed.

`1` Both error messages and warnings are displayed (default).

`>1` Outputs diagnostic information which may not be significant for the user.

Arguments

`tag` This argument is used to specify the value in the first column of `termtab`; a verified line is then only selected if it matches `tag`. In this case, the last field (a command) is executed. The default entry in this column is either "start" or "stop"; this entry is selected when starting or stopping the system using `rc` scripts (see below).

hostname

Specifies the name of the host system to which the device is connected via TACLAN (value of the second column in `termtab`). The verified line must match this host system. If both the first and the second column match, the line is selected and the last column is executed.

devname

Specifies the name of the device (value of the third column in `termtab`), which a verified line must match. If the first three columns match, the line is selected and the last column is executed.

portno

Specifies the port number on the TACLAN (fourth column in `termtab`) that a verified line must match. If the first four columns match, the line is selected and the last column is executed.

devtype

Specifies the type of device (fifth column in `termtab`) which a verified line must match. If the first five columns match, the line is selected and the last column is executed.

NOTES

The sixth column of `termtab` contains internal information. The seventh column contains information on a secondary device that may be connected. The last column contains special non UNIX commands which are used to terminate a daemon process, for example.

The following commands are supported:

`KILLMUX`
: Ends the `mux` daemon

`KILLRTELNET`
: Ends the `rtelnet` daemon

`KILLTSVR_D`
: Ends the `tsvr_d` daemon

`UNLINKDPTG`
: Deactivates the `dptg` administration channel

`termadm` is linked to the `rc2.d` and `rcS.d` directory so that it can be used to start or terminate devices using `init`. The first column of `termtab` table therefore contains the entries "start" and "stop". `termadm` can also be used by the system administration to start or terminate devices which are connected to a terminal server that is to be replaced; it can also be used to activate a connection to another host which is always accessed if the primary host is not active. The first column of the `termtab` file also has the value "backup".

DIAGNOSTICS

All messages are output to `stderr`. The following is a list of return values:

`0` = Every command was executed successfully.

`1` = No lines match the specified arguments.

`2` = Syntax error.

`3` = An error occurred when executing the command(s).

`4` = Invalid option in the command line.

EXAMPLES

Let us assume that the table has the following contents:

```
start:myannex:myterm:1:ba80:myannex@1::/opt/bin/dptg2/tsvr_d -a ...
stop:myannex:myterm:1:ba80:myannex@1::KILLTSVR_D
start:brother:small:1:ba80:brother@1::/opt/bin/dptg2/tsvr_d -a ...
stop:brother:small:1:ba80:brother@1::KILLTSVR_D
start::tiny:1:ba80:itp0,1::/opt/bin/dptg2/tsvr_d -a ...
stop::tiny:1:ba80:itp0,1::UNLINKDPTG
start:brother:failsafe:0:97801-5:::rtelnet ...
stop:brother:failsafe:0:97801-5:::KILLRTELNET
```

The following `termadm` calls are then correct:

```
termadm start myannex
```
(starts all TACLAN "myannex" devices)
```
termadm stop ERROR*ERROR myterm
```
(stops the "myterm" device)
```
termadm start ERROR*ERROR ERROR*ERROR ba80
```
(starts all ba80 terminals)

LOCALE

The `LC_MESSAGES` environment variable governs the language in which message texts are displayed.

If `LC_MESSAGES` is undefined or is defined as the null string, it defaults to the value of `LANG`. If `LANG` is

likewise undefined or null, the system acts as if it were not internationalized.

If any of the locale variables has an invalid value, the system acts as if none of the variables was set.

The `LC_ALL` environment variable governs the entire locale. `LC_ALL` takes precedence over all the other environment variables that affect internationalization.

FILES

`/var/sadm/terms/termtab`

SEE ALSO

`ksh`(1), `init`(1M), `termadd`(1M), `termdel`(1M), `termtab`(4).

System Administrator's Reference Manual

termdel(1M)

NAME

`termdel -` remove a device configured with termadd

SYNOPSIS

`termdel −F` *config-file*

`termdel` [`−T` *type*] [`−v` *verbose*] *name1* [*name2*] ... [`!`]

DESCRIPTION

`termdel` removes the configuration of the devices *name*, *name2*, ..., which (this can also be a serial device, printer or PC), is specified in the command line. This task can then be carried out easily using `termdel` itself, if the device is not (no longer) configured correctly. It removes the corresponding definitions, if they have been entered in the system. The `−T` option, however, should be specified.

A special algorithm prevents `termdel` being executed simultaneously with `termadd` [see `termadd(1M)`]. If either `termadd` or `termdel`, however, has been terminated by a system error, it may not be possible to activate `termdel`. This can be eliminated by specifying an exclamation mark ("`!`") as the last parameter in the command line for the next call.

`termdel` can be called in two ways. The devices to be removed can be specified using the call line, and the other option is to call it using a file (`F` option). The devices are specified in a `:` notation here. These calls are generally initiated by configuration generators such as `Config`.

`termdel` does not work on the basis of transactions. Where possible, `termdel` removes all the necessary system entries for a configuration. The node with the device name is deleted last – if no serious error occurred up to that point. `termdel` can therefore also be called several times in succession in the case of errors.

OPTIONS

name1, *name2*, ...

Specified the device name relative to `/dev/term`. If placeholders are specified as device names, they must be enclosed in single quotation marks (`'`).

For consistency reasons, no nodes that were created internally by `termadd` can be removed as device names, i.e. only nodes that were configured using the `n` option for `termadd` are permitted as device names.

If the name of the 0 channel on a BA80 terminal is specified, all the associated windows are also deleted.The secondary devices (such as printers), however, must have been deleted beforehand.

`−v` *verbose*

Controls the output of diagnostic information. Values for *verbose*:

`0` Only error messages are displayed.

`1` Both error messages and warnings are displayed (default).

`>1` Outputs diagnostic information that may not be significant for the user.

`−T` *type*

`termdel` can recognize the device type from a SAF file. If this file has been corrupted or deleted, the type can also be specified using this option, in which case the type indicated here must always match the type you used for configuration with `termadd`.

NOTES

Only the superuser can execute `termdel`.

DIAGNOSTICS

All messages are output to `stderr`. The value 0 is output once the command has been successfully

executed, while a value other than 0 is output following an error. When the command is executed, 0 is output. If the device to be deleted has not been configured, the value 0 is still output. If an error occurs, a value other than 0 is output.

`1` = Error deleting, see `stderr`.

`2` = Syntax error.

EXAMPLES

```
termdel tty4711
termdel 'sales*'
termdel –v2 scrawl cranes
termdel –Tba80 curve
```

LOCALE

The `LC_MESSAGES` environment variable governs the language in which message texts are displayed.

If `LC_MESSAGES` is undefined or is defined as the null string, it defaults to the value of `LANG`. If `LANG` is likewise undefined or null, the system acts as if it were not internationalized.

If any of the locale variables has an invalid value, the system acts as if none of the variables was set.

The `LC_ALL` environment variable governs the entire locale. `LC_ALL` takes precedence over all the other environment variables that affect internationalization.

FILES

```
/dev/term
/etc/ttydefs
/etc/hdlcdefs
/etc/ttBtype
/etc/termtab
/usr/bin/terms/crontab
/var/sadm/terms/termtab
/var/sadm/terms/choices
/var/sadm/terms/hdlclabs
/var/sadm/terms/rtydefs
```

SEE ALSO

`getopts(1)`, `login(1)`, `intro_sarm1(1M)`, `sac(1M)`, `termadd(1M)`, `termadm(1M)`, `termshow(1M)`, `ttymon(1M)`.

System Administrator's Reference Manual

termshow(1M)

NAME

`termshow -` show device configuration

SYNOPSIS

`termshow` [–T *type*] [*show_options*] [*name ...*]

`termshow –?`

DESCRIPTION

`termshow` searches for devices that have been configured (i.e. serial devices, printers or PCs), in `/dev/term` and displays their attributes using a suitable `termadd` command line. The entries in `/dev/term` are output in the order generated by `ls`. This overview does not represent the order in which the `termadd` commands were processed, when this configuration was created. The arguments displayed depend on the respective "intelligence" level, which can be specified by the user (see below). In principle, there are various ways of configuring devices, especially terminals in UNIX. `termshow` only recognizes devices that have been defined under SAF (Service Access Facility). You may not get any (correct) information on `init/getty` configurations.

`termshow` can handle arguments with placeholders. This allows the user to keep the number of names to be checked as low as possible and to speed up processing, as only the devices actually required are checked.

The placeholders are evaluated by the shell in the current directory. The user must therefore enclose the placeholders in single quotation marks (') so that they can be evaluated at any time with `termshow` in the directory `/dev/term`.

Each argument specified represents a restriction, which must be ANDed to the other arguments. This means that information is only output on a device if this matches all the arguments (including placeholders) in the command line.

An argument containing placeholders must be enclosed in single quotation marks ('); this applies in particular for the character `*` (see **EXAMPLES**).

`termshow` has no memory, i.e. the complete configuration of a device is checked again every time it is called. `termshow` output therefore gives a current indication of the present system status every time. In the case of inconsistent configurations, corresponding warnings are also output. Approximately 0.5 s are required per device with a low computer load, in order to check the complete configuration.

ARGUMENTS

All options and arguments must be specified as described under `intro_sarm1(1M)` . Please ensure, in particular, that the arguments of options do not contain blanks [see also `getopts(1)`].

`termshow` checks either the entries in `/dev/term`, which have been specified in the command line, or *all* entries (if there are no corresponding arguments in the command line). These arguments can contain placeholders in any format defined under `ksh` (e.g. `tty`[1-3]*). `termshow` outputs the relevant attributes in `termadd` format (see below).

–T *type*

Specifies the type of the device for which information is to be output. The arguments can contain placeholders in any `ksh` format (e.g. `97801-`[34] or `ba80*`). Each placeholder is evaluated in relation to the device types supported by `termadd` on this system.

The following specifications are possible with *show_options*:

–m *Mdevno,mdevno*

The requested device configuration is only output for the specified device number. The arguments are interpreted as *Mdevno* (device class) and *mdevno* (device number).

The arguments should be separated from each other by a comma (","), blanks are not permitted. The

argument can contain placeholders in any `ksh` format (e.g. `'20[0-3],*'`).

`-r` *hostname,portno*
Specifies the remote connections for which information is to be output. *hostname* specifies the name of the terminal server; *portno* indicates the number of the port to which the device is connected.

The argument can contain placeholders in any `ksh` format (e.g. `annex[0-9],[0-9][01]`).

The arguments should be separated from each other by a comma (","), blanks are not permitted.

`-b` *boardname,portno*
Specifies the direct/local connections for which information is to be output. *boardname* indicates the name and number of the controller; *portno* specifies the number of the port to which the terminal is connected.

Either *boardname* or *portno* can contain numbers. A `*` (for each number permitted) can also be used instead of these numbers.

The arguments *boardname* and *portno* should be separated from each other by a comma (","), blanks are not permitted. The argument can contain placeholders in any `ksh` format (e.g. `'20[0-3],*'`). The syntax meets the requirements of the `-b` option of `termadd`.

`-i` *intelligence*
Controls the level of processing required to output the configuration details for the port (`m`, `b`, or `r` options of `termadd`).

If *intelligence* equals 1, then only the attributes *terminalname*, *Mdevno*, and *mdevno* are output. No option is permitted with this value.

If *intelligence* equals 2, `termshow` also tries to output the `termadd` options `-T`, `-n`, `-t`, `-l`, `-p`, `-s`, and `-d`. The connection point to the host is indicated using the `-m` option. The `-m` option is suitable for locally or via Ethernet connected devices [see `termadd(1M)`], but the `-m` option displayed cannot be used for a `termadd` command by devices configured with the `-r` option via RTELNET.

If *intelligence* equals 3 (default), `termshow` tries to determine the connection point using the `termadd` option `-b`, `-r`, or `-c`. The information on the device type, which is stored in the configuration file for the port monitor (`_pmtab`), must not be deleted for this type of output.

`termshow` outputs additional information on special options, such as the `muxid` and `channel` of a ba80-type terminal.

Even with this degree of *intelligence*, `termshow` cannot report accurately on all devices, especially if they have not been configured with `termadd`, rather with some "free-style" algorithm. To output information on these devices, *intelligence* must be set to 4. Configurations may exist, however, that also cannot be recognized in full. All devices and device attributes that cannot be recognized with 100% reliability are output with a corresponding message to `stderr`.

`-t` *pmtag*
Specifies the monitor used to provide the LOGIN. The argument can contain placeholders in any `ksh` format (e.g. `itp`[01]). The placeholders are evaluated in `/etc/saf`.

`-l` *linesetting*
Specifies the flag in `/etc/ttydefs`, which is assigned to the relevant terminals.

The argument can contain placeholders in any `ksh` format (e.g. `ba80*8`).

`-k` *keyboard*
Specifies the type of keyboard to be used by the particular terminal.

The argument can contain placeholders in any `ksh` format (e.g. `??0`[68]).

`-s` *nationality*
Specifies the language, i.e. a special keyboard assignment. Restrictions on this option are only possible if *intelligence* is set to at least 3 (default). The argument can contain placeholders in any `ksh` format (e.g. `ch.?`).

`-d` Disables flag: Information is only output for devices, especially terminals with no login service.

–v *verbose*

Controls diagnostic output. Values for *verbose*:

`0` Only error messages are displayed.

`1` Both error messages and warnings are displayed (default).

`2` Messages are output on the processing status in relation to the overall processing being carried out, if 10% more devices are being examined.

`>2` Outputs diagnostic information that may not be significant for the user.

`name` Is a device name (e.g. `'tty*'`). *Note:* Specifications with placeholders *not* enclosed in single quotation marks are evaluated in the current directory, while specifications with placeholders *in* single quotes are evaluated in `/dev/term`.

`termshow –?`

Outputs help information.

NOTES

To correctly output the information on the devices, it must be possible to access the device type to be checked. This type is stored by `termadd` in the configuration file of the controlling monitor. The information is marked `Do never change: TYPE=`*string* !. It is strongly recommended not to modify this information.

The `–f` option is also available and can be used to switch the output format to an internal format. There is also a `–C` option. Both options are only used by `Config` (see `sysadm`(1M) , menu item "Configuration").

With `termshow`, device configurations can be stored or duplicated, as the output can be reused again immediately. The logical sequence must be produced beforehand, however, so that the channels with the number 0 are always handled first.

`termshow` is also suitable for checking the configuration of "hanging" terminals. If the configuration is consistent, you can buffer the output, delete the terminal using `termdel`, and then configure the `termshow` output again.

DIAGNOSTICS

The command always terminates with the value 0, provided that no internal error occurred. If an internal error occurred, the value 1 is returned.

EXAMPLES

The following examples illustrate among other things the use of placeholders and options that can restrict the command execution to the devices, about which information is required.

`termshow cranes`

Only displays the device with the name "cranes"; the device class and the device number have been converted to the corresponding board type.

`termshow –T '978*' –i1` [a-z]*

Displays all terminals whose names start with a lowercase letter and whose type starts with "978". The connection is not specified symbolically, rather as the device class and device name.

`termshow –T '978*' –b 'itp`[23],*'

Displays all terminals that are connected to board itp2 or itp3.

`termshow –T 'ba8*' –m '201,??'`[12][0-9]?

Displays all terminals that have the device class 201 and a name between 10 and 29.

`termshow –T 97801-vt –t itp0`

Displays all terminals that are of type 97801-vt and whose `tty` monitor has the identifier itp0.

`termshow –T '97801-*'`

Displays all terminals whose type starts with the string 97801.

`termshow –T'serial*' –b'sih3,sw8rV3,1*([0-9])'`

Displays all devices that are connected to the V.24 String 3, have a device address between 1 and 999, and are of the type "serial".

LOCALE

The `LC_MESSAGES` environment variable governs the language in which message texts are displayed.

If `LC_MESSAGES` is undefined or is defined as the null string, it defaults to the value of `LANG`. If `LANG` is likewise undefined or null, the system acts as if it were not internationalized.

If any of the locale variables has an invalid value, the system acts as if none of the variables was set.

The `LC_ALL` environment variable governs the entire locale. `LC_ALL` takes precedence over all the other environment variables that affect internationalization.

FILES

`/dev/term`
`/etc/ttydefs`
`/etc/ttytype`
`/etc/hdlcdefs`
`/etc/keytables`
`/etc/hosts`
`/opt/xb`
`/etc/saf/`*pmtag*`/_pmtab`
`/var/sadm/terms/termtab`
`/var/sadm/terms/choices`
`/var/sadm/terms/hdlclabs`

SEE ALSO

`getopts`(1), `ksh`(1), `intro_sarm1`(1M), `sac`(1M), `termadd`(1M), `termadm`(1M), `termdel`(1M), `ttymon`(1M).

On `Config`:

Manual *System Administration and Hardware Configuration Using the SYSADM User Interface*.

System Administrator's Reference Manual

testecc(1M)
RM600 only

NAME

`testecc -` log CPU board cache and memory board errors

SYNOPSIS

`/sbin/testecc` [`–m` *threshold_value*] [`–c` *threshold_value*] [`–t` *threshold_value*] [`–b` *cpu_no*]

DESCRIPTION

`testecc` ensures continuous checking of the memory boards and CPU boards. The number of single-bit and multi-bit errors on all memory boards and the number of single-bit errors in the caches on the CPU boards are checked. If the threshold value is exceeded, defined log entries are made by `testecc`. Event Management handles the filtering of these entries, and the response script assigned to the event is responsible for identifying the faulty module. The information on the boards is reset after the query.

OPTIONS

`–m` *threshold_value*
: *threshold_value* defines the number of non-critical single-bit errors permitted on a memory board within a specific period of time (time between 2 calls). If the set value is exceeded, logbook entries are made (Level: `WARNING`) and the relevant memory slot is flagged for LAR. If the threshold value is not exceeded, any errors that occur are logged (Level: `NOTICE`), but are not flagged for LAR. If the parameter is not specified, 0 is assumed.

`–c` *threshold_value*
: As with the `–m` option, this parameter determines the number of cache errors that should be tolerated on a CPU board. This threshold value is only valid for a CPU board with online memory.

`–t` *threshold_value*
: Identical to the `–c` option, but the threshold value only relates to the R10000-CPU (RM600-E20/60).

`–b` *cpu_no*
: In the case of calls where memory boards must be addressed with an agent ID greater than 16, the response of these memory boards is incorrect (firmware problem). In order to prevent this happening, the process is linked to the CPU 0. With the `–b` option, the process can be linked to another CPU with the number *cpu_no*. The default for *cpu_no* is 0.

Integration in the system

An entry with the following format should be made in the `var/spool/cron/crontabs/root` file:

```
0 0,4,8,12,16,20  * * * /sbin/testecc -m 3 -c 3 -t 3
```

This means that the error information is read every 4 hours. Since the threshold values are set to 3, the module is flagged for LAR if there have been more than three errors since the last call.

Log file entries

Error messages are output to the LOG3 server. If a single-bit or multi-bit error occurs on a memory board, the complete data for each of these marked boards is transferred to the server. This complete information is also output in the case of cache error information for CPU boards.

DIAGNOSTICS

General:

`–1` The call parameters are incorrect.

Program part for logging memory board errors:

`–2` A`malloc()` system call failed.

`-3` This value is returned if at least one CPU or one memory board has a firmware release installed that is too old.

`-4` The number of banks on a memory board was recognized as invalid.

`-5` The `log3write()` function returned an error.

`>0` An error occurred in the `sysmips()` system call (reading out the boards). The return value for `errno` is output.

Program part for logging cache errors:

`-2`, `-5`, `>0`
As above.

These errors are multiplied by 100 and added to the error code for logging memory board errors. This means that errors can be evaluated in both program parts.

System Administrator's Reference Manual

tic(1M)

NAME

`tic` - terminfo compiler

SYNOPSIS

`tic` [–v[*n*]] [–c] *file*

DESCRIPTION

The `tic` command translates a `terminfo` file from source format into compiled format. The results are placed in the `/usr/share/lib/terminfo` directory. The compiled format is necessary for use with the library routines in `curses(3X)` .

OPTIONS

–v*n* Specifies that (verbose) output be written to standard error trace information showing `tic`'s progress. The optional integer *n* is a number from 1 to 10, inclusive, indicating the desired level of detail of information. If *n* is omitted, the default level is 1. If *n* is specified and greater than 1, the level of detail is increased.

–c Specifies to check only *file* for errors. Errors in `use=` links are not detected.

file Contains one or more `terminfo` terminal descriptions in source format [see `terminfo(4)`]. Each description in the file describes the capabilities of a particular terminal. When a `use=`*entry-name* field is discovered in a terminal entry currently being compiled, `tic` reads in the binary from `/usr/share/lib/terminfo` to complete the entry. (Entries created from *file* will be used first. If the environment variable `TERMINFO` is set, that directory is searched instead of `/usr/share/lib/terminfo`.) `tic` duplicates the capabilities in *entry-name* for the current entry, with the exception of those capabilities that explicitly are defined in the current entry.

If the environment variable `TERMINFO` is set, the compiled results are placed there instead of `/usr/share/lib/terminfo`.

Total compiled entries cannot exceed 4096 bytes. The name field cannot exceed 128 bytes. Terminal names exceeding 14 characters will be truncated to 14 characters and a warning message will be printed.

NOTES

When an entry, such as `entry_name_1`, contains a `use=`*entry_name_2* field, any canceled capabilities in *entry_name_2* must also appear in `entry_name_1` before `use=` for these capabilities to be canceled in `entry_name_1`.

FILES

`/usr/share/lib/terminfo/?/*`
Compiled terminal description database

SEE ALSO

`captoinfo(1M)` , `infocmp(1M)` , `curses(3X)` , `terminfo(4)` .

System Administrator's Reference Manual

tosmtp(1M)

NAME

`tosmtp -` send mail to SMTP

SYNOPSIS

`/usr/lib/mail/surrcmd/tosmtp` [`–f`] [`–n`] [`–u`] [`–d` *domain*] [`–H` *helohost*] *sender host recip ...*

DESCRIPTION

`tosmtp` translates a Reliant UNIX System mail message (read from standard input) into an RFC822 mail message, which can then be delivered with SMTP. `tosmtp` is normally invoked by `smtpqer` as part of the process of queuing mail for delivery.

OPTIONS

`–d` *domain*
: Passes the specified *domain* directly to the `smtp` program.

`–f`
: Acts as a filter. The RFC822 message is sent to standard output.

`–H` *helohost*
: This option can be used to specify the name to be used for the host in the initial `SMTP HELO` message. This option is also passed to the `smtp` program.

`–n`
: Do not place a `To:` line in the resulting RFC822 header.

`–u`
: Do no conversion. Standard input is sent directly to standard output.

FILES

`/usr/lib/mail/surrcmd/smtp`
: Where the message is piped to

SEE ALSO

`smtp`(1M) , `smtpqer`(1M) .

RFC822 – Standard for the Format of ARPA Internet Text Messages.

System Administrator's Reference Manual

tpstat(1M)

NAME

`tpstat -` report magnetic tape I/O statistics

SYNOPSIS

`tpstat` [`–abdgoqrvx`] [*tpname ...*]

`tpstat –z` [*tpname ...*]

`tpstat` *interval* [*count*] [*tpname ...*]

`tpstat –L` *interval* [*count*] [*tpname ...*]

DESCRIPTION

`tpstat` outputs the I/O statistics collected for tape drives since the system was loaded or since the last time the I/O statistic was reset.

Scaled profiles can be displayed for the block size and the response time as well as general details about average utilization and total usage.

Current tape usage can be monitored by specifying an *interval* in seconds. A continuous display of average usage over each *interval* is produced.

You must select an option or specify an *interval*. `tpstat` will not work correctly without an option or *interval*.

Statistics can be cleared with the `–z` flag. If you use the `–z` option with other options, the other options will be ignored; therefore, always use the `–z` option on a separate line.

OPTIONS

`–a` All statistical profiles.

`–b` A profile of the number of blocks that were requested in each I/O.

`–d` Profile of the response time in milliseconds from the sending of the I/O to the device and the receipt of the completion interrupt.

`–g` Read, write and byte counts. The read and write counts are the total number of requests of each type. The byte count is the total number of bytes transferred in both read and write requests.

`–q` Profile of the tape request queue length; The value is always 1 for tape devices, since there is no queue as with disks. This value is only maintained for compatibility reasons [cf. `sar(1)`].

`–r` Profile of the response time in milliseconds from the sending of the I/O to the device and the receipt of the completion interrupt.

The `–r` option corresponds to the `–d` option.

`–v` Forces the display of all data. If the `–v` flag is not used, `tpstat` will truncate trailing zeros and print ellipses at the end of non-zero data.

`–x` Prints a summary of the options.

`–z` Clears all statistics.

`–L` Display load average over interval.

The optional trailing *interval* causes `tpstat` to report the average tape transfers per second on each tape over that *interval* repeated an optional *count* times. If *count* is omitted, the output is repeated continuously. Drives may be specified with an optional trailing *tpname*. Names are given in either the 13 character device name form or an abbreviated form which includes bus number, bus type, and controller and device numbers. For example, `ios0/stape001` or `s001`. The shell metacharacter syntax "`?`", "`*`", and "`[`" may also be used. For example, `ios0/stape* ?tape[01]??`, or even `*[01]`. These names should be escaped on shell level where necessary.

If no drives are specified, then the five tapes with the highest usage since boot will be displayed. If the `-L` load average flag is specified, then the four tapes with highest usage during the interval are displayed.

Profile information is collected from several arrays within the kernel address space. Each array has two logical areas with selectable scaling of data. These arrays are described by maximum values of both segments and the scale factor within both segments.

Averages are generated which are based on scaling data with the beginning value of the sample cell or the end value of the sample cell. The true average can be extrapolated between these values. Because the data is collected in cells, the exact average cannot be determined.

The load average is a number indicating the overall effect of a tape on system I/O performance. While the details of I/O performance may be obtained by carefully examining the profile information, the load average is meant to give an overview of performance of each device in a single number. The tape load average is calculated as an exponential reduction of the current load. The load is calculated for an interval using the following formula:

```
load = device_busy_time * ( L1 + L2 )
```

where:

```
L1 = new_requests/requests_completed
L2 = average_queue_depth
```

SEE ALSO

sar(1), mpstat(1M), vmstat(1M).

System Administrator's Reference Manual

troff(1M)

NAME

`troff` - text formatting and typesetting

SYNOPSIS

`troff` [*option* ...] [*file* ...]

DESCRIPTION

`troff` formats text in the named *file* for printing on a laser printer or phototypesetter.

If no *file* argument is present, standard input is read. An argument consisting of a single minus (–) is taken to be a file name corresponding to standard input.

OPTIONS

The options, which may appear in any order so long as they appear before the files, are:

–o*list*
: Prints pages in the comma-separated *list* of numbers and ranges. A range *N* – *M* means *N* through *M*; initial –*M* means up to *M*; final *N*– means from *N* to the end.

–n*N* Number first generated page *N*.

–m*name*
: Prepends the macro file `/usr/lib/tmac/tmac.`*name* to the input *files*.

–r*aN* Sets register *a* (one character name) to *N*.

–i Reads standard input after the input files are exhausted.

–q Invokes the simultaneous input-output mode of the `.rd` request.

–z Produces no output: diagnostics and `.tm` messages only.

–a Sends a printable ASCII approximation of the results to standard output.

–T*name*
: Prepares output for typesetter *name*. The default *name* is `post`.

FILES

`/usr/lib/tmac/tmac.*`
: standard macro files

`/usr/lib/font/dev*/*`
: font width tables

`/usr/tmp/trtmp*`
: temporary file

SEE ALSO

`dpost`(1M), `eqn`(1M), `grap`(1M), `nroff`(1M), `pic`(1M), `tbl`(1M), `font`(5), `troff`(5).

REFERENCES

J. F. Ossanna, *Nroff/Troff User's Guide*

B. W. Kernighan, *A Typesetter-Independent Troff*

B. W. Kernighan, *A Troff Tutorial*

D. Dougherty, T. O'Reilly, *UNIX TEXT PROCESSING*, Hayden Books, Indianapolis, Indiana 46268 USA, ISBN 0-672-46291-5

System Administrator's Reference Manual

tsvrget(1M)

NAME

`tsvrget -` query network station

SYNOPSIS

`tsvrset` [`–T` *type*] `–l` *label-family* [*options*] *netstation*

DESCRIPTION

This command queries the configuration of a terminal server or printer server and outputs it to `stdout`. The SNMP protocol is used in this case. As a prerequisite, the name or the Internet address of the terminal server or printer server must be defined (*netstation*).

Internet addresses are only converted to names if the `n` option was not selected, there is a suitable entry in `/etc/hosts`, and the attribute has a "HOSTNAME" name part.

OPTIONS

`–T` *type*
: Specifies the type of terminal server or printer server. The default is "annex8". To ensure consistent configuration, there must be a directory of the same name in `/var/sadm/terms/tsvrtypes`. The corresponding attribute sets, among other things, are recorded here [see `mibtool(1M)`].

`–l` *label-family*
: Defines the attribute sets to be used. A collective name (*label-family*) can consist of up to 9 individual attribute set names. Each attribute set must be defined in one of the three files with the ending ".def" referred to below (see **FILES**) in the `/var/sadm/terms/tsvrtypes/`*Type* directory. The definition of an attribute set complies with `mibtool(1M)` . A collective name has the following format:

  label[:*label*][:*label*][:*label*][:*label*][:*label*][:*label*][:*label*][:*label*]

`–p` *portno*
: Defines the port number to be configured. The attributes of the selected attribute set, which have the ending ".i", are updated in accordance with this number.

`–v` [`1-5`]
: Defines the scope of the control output in different levels.

`–n`
: Switches off conversion of addresses to names.

`–f` *file*
: Defines the output file to which the results are written.

`–t` *dchar*
: Defines the field separator that is to be inserted between the attribute name and its type.

EXAMPLES

```
tsvrget -l debug -p5 myannex
tsvrget -T annex8 -l port:slave annex5
tsvrget -n -l base:routing newannex
```

FILES

`/var/sadm/terms/tsvrtypes/var/sadm/terms/tsvrtypes/`*Type*`/snmpinfo.dat/var/sadm/terms/tsvrtypes/`*Type*`/config.def/var/sadm/terms/tsvrtypes/`*Type*`/misc.def/var/sadm/terms/tsvrtypes/`*Type*`/add-on.def`

SEE ALSO

`mibtool(1M)`, `tsvrset(1M)`.

System Administrator's Reference Manual

tsvrset(1M)

NAME

`tsvrset -` configure network station

SYNOPSIS

`tsvrset` [`–T` *type*] `–l` *label-family* [*options*] *netstation* [*assignments*]

DESCRIPTION

This command configures a terminal server or printer server and outputs – if the `–d` option is specified – the settings to `stdout`. The SNMPv2 protocol is used in this case. As a prerequisite, the name or the Internet address of the terminal or printer server must be defined (*netstation*).

Attribute values that are not defined in the corresponding attribute set, but are instead defined as a variable, can be assigned current values via the command line (*assignments*). Only variables comprising uppercase letters must be used. The correct syntax complies with `ksh(1)` . This means that a character string "*name=value*" must not be separated by blanks.

All queries to a terminal server or printer server are only processed by the device if it belongs to the same "community" as the system which made the query. As long as there are no SNMPv2 agents, the community identifier "`public`" is used.

The command logs each call in the `/var/sadm/terms/netdefs` file. There is precisely one entry with "*labels*:*assignments*" for each combination of "*netstation*:*type*:*port_number*". This log file constantly increases in size and is not updated when a terminal server or printer server is deleted.

OPTIONS

`–T` *type*
: Specifies the type of terminal server or printer server. The default is "annex8". To ensure consistent configuration, there must be a directory of the same name in `/var/sadm/terms/tsvrtypes`. The corresponding attribute sets, among other things, are recorded here [see `mibtool(1M)`].

`–l` *label-family*
: Defines the attribute sets to be used. A collective name (*label-family*) can consist of up to 9 individual attribute set names. Each attribute set must be defined in one of the three files with the ending ".def" referred to below (see **FILES**) in the `/var/sadm/terms/tsvrtypes/`*Type* directory. The definition of an attribute set complies with `mibtool(1M)` . A collective name has the following format:

  label[:*label*][:*label*][:*label*][:*label*][:*label*][:*label*][:*label*][:*label*]

`–p` *portno*
: Defines the port number to be configured. The attributes of the selected attribute set, which have the ending ".i", are updated in accordance with this number.

`–v` [`1-5`]
: Defines the scope of the control output at different levels.

`–d` Outputs the attribute values after the command is executed.

assignments
: "assignments" have the following structure:

  variable_name={*value* | "*value_with_blanks*"}

  There must be no blanks to the left and right of the "=" character.

EXAMPLES

```
tsvrset -Tannex8 -p31 -lslave:19200np myannex
tsvrset -Tannex8 -p31 -lslave:19200np -d myannex
tsvrset -l port:slave myannex \
```

```
HOSTNAME=1.2.3.4 PORTNO=7411 COMMENT="for_test"
```

FILES

/var/sadm/terms/tsvrtypes
/var/sadm/terms/tsvrtypes/*Type*/snmpinfo.dat
/var/sadm/terms/tsvrtypes/*Type*/config.def
/var/sadm/terms/tsvrtypes/*Type*/misc.def
/var/sadm/terms/tsvrtypes/*Type*/add-on.def
/var/sadm/terms/netdefs

SEE ALSO

mibtool(1M), tsvrget(1M).

System Administrator's Reference Manual

tt_open(1M)

NAME

`tt_open -` background program for simple V24 tty printer ports

SYNOPSIS

`/usr/bin/tt_open` *full_tty_device_names*

DESCRIPTION

Simple tty printer ports (V24) are restricted by the fact that the flow control (^S/^Q) does not function properly, depending on the quick opening/closing of the tty port.

`tt_open` is a small program, which permanently opens the specified printer tty ports. The program is started by root during system startup in the background and runs indefinitely.

EXAMPLES

```
/usr/bin/tt_open /dev/term/0 /dev/term/1 ... &
```

or

```
nohup /usr/bin/tt_open `cat print_ttys` &
```

`tt_open` permanently opens the specified tty ports, flow control (^S/^Q) now works correctly for these simple printer ports (V24). `print_ttys` is a file, which lists the complete tty printer port names (`/dev/...`) on separate lines.

NOTES

Do not use this program if IOCS is installed for the ports.

Only the superuser is allowed to execute `tt_open`.

FILES

`/etc/rc2.d/S19tt_open`

System Administrator's Reference Manual

ttyadm(1M)

NAME

`ttyadm -` format and output port monitor-specific information

SYNOPSIS

`/usr/sbin/ttyadm` [`-b`] [`-c`] [`-k`] [`-r` *count*] [`-h`] [`-i` *msg*] [`-m` *modules*] [`-p` *prompt*] [`-t` *timeout*] `-d` *device* `-l` *ttylabel* `-s` *service*

`/usr/sbin/ttyadm -S` [`-i` *svctag*] *pmtag*

`/usr/sbin/ttyadm -T` *level pmtag*

`/usr/sbin/ttyadm -V`

DESCRIPTION

The `ttyadm` command is an administrative command that formats `ttymon`-specific information and writes it to standard output. The Service Access Facility (SAF) requires each port monitor to provide such a command. Note that the port monitor administrative file is updated by the Service Access Controller's administrative commands, `sacadm` and `pmadm`. `ttyadm` provides a means of presenting formatted port monitor-specific (i.e. `ttymon`-specific) data to these commands.

Furthermore, `ttyadm` can be used to output the status of the connections monitored by `ttymon`, and to activate or deactivate trace mode.

Format output

`-b` Sets the "bidirectional port" flag. When this flag is set, the line can be used in both directions. `ttymon` will allow users to connect to the service associated with the port, but if the port is free, `uucico`, `cu`, or `ct` can use it for dialing out.

`-c` Sets the connect-on-carrier flag for the port. If the `-c` flag is set, `ttymon` will invoke the port's associated service immediately when a connect indication is received (i.e. no prompt is printed and no baud-rate searching is done).

`-k` Sets the "pre-prompt-config" for the connection. This determines that the configuration file of the service is interpreted before the prompt is written. Otherwise, this file is interpreted before the service is started.

`-d` *device*
device is the full pathname of the device file for the TTY port.

`-h` Sets the hangup flag for the port. If the `-h` flag is not set, `ttymon` will force a hangup on the line by setting the speed to zero before setting the speed to the default or specified value.

`-i` *message*
Specifies the inactive (disabled) response message. This message will be sent to the TTY port if the port is disabled or the `ttymon` monitoring the port is disabled.

`-l` *ttylabel*
Specifies which *ttylabel* in the `/etc/ttydefs` file to use as the starting point when searching for the proper baud rate.

`-m` *modules*
Specifies a list of pushable STREAMS modules. The modules will be pushed, in the order in which they are specified, before the service is invoked. *modules* must be a comma-separated list of modules, with no white space included. Any modules currently on the stream will be popped before these modules are pushed.

`-r` *count*
When the `-r` option is invoked, `ttymon` will wait until it receives data from the port before it displays a prompt. If *count* is equal to zero, `ttymon` will wait until it receives any character. If *count* is greater than

zero, ttymon will wait until *count* newlines have been received.

−p *prompt*
Specifies the prompt message, e.g. "login: ".

−s *service*
service is the full pathname of the service to be invoked when a connection request is received. If arguments are required, the command and its arguments must be enclosed in double quotes. If the pseudo search NONE is specified, ttymon does not start a service. The connection is simply kept open.

−t *timeout*
Specifies that ttymon should close a port if the open on the port succeeds and no input data is received in *timeout* seconds.

−V Displays the version number of the current /usr/lib/saf/ttymon command.

Trace (option −T)

This option can be used to activate and deactivate output of trace information. Trace output is included in the log file of ttymon.

−T *level*
level = 1 activates the trace, while *level* = 0 deactivates it again.

pmtag
The name of the ttymon under SAF.

Status output (option −S)

Attymon of the status reached is written to stdout for each service (*svctag*). As a side-effect, the status and _pmtab data for each service is written to /var/saf/*pmtag*/State.

−i *svctag*
If this option is used, only the status of the specified *svctag* is output. In addition to the status, the _pmtab data of the *svctag* service is also output.

pmtag
The name of the ttymon under SAF for which the specifications are to be made.

DIAGNOSTICS

If successful, ttyadm will generate the requested information, write it to standard output, and exit with a status of 0. If ttyadm is invoked with an invalid number of arguments or invalid arguments, or if an incomplete option is specified, an error message will be written to standard error and ttymon will exit with a non-zero status.

FILES

/etc/ttydefs

SEE ALSO

pmadm(1M) , sacadm(1M) , ttymon(1M) .

System Administrator's Reference Manual

ttymon(1M)

NAME

`ttymon` - port monitor for terminal ports

SYNOPSIS

`/usr/lib/saf/ttymon`

`/usr/lib/saf/ttymon` `-g` [`-d` *device*] [`-h`] [`-t` *timeout*] [`-l` *ttylabel*] [`-p` *prompt*] [`-m` *modules*]

DESCRIPTION

`ttymon` is a STREAMS-based TTY port monitor. Its function is to monitor ports, to set terminal modes, baud rates, and STREAMS modules for ports, and to connect users or applications to services associated with the ports. Normally, `ttymon` is configured to run under the Service Access Controller (SAC), as part of the Service Access Facility (SAF). It is configured using the `sacadm(1M)` command. Each instance of `ttymon` can monitor multiple ports. The ports monitored by an instance of `ttymon` are specified in the port monitor's administrative file. The administrative file is configured using the `pmadm(1M)` and `ttyadm(1M)` commands. When an instance of `ttymon` is invoked by the `sac(1M)` command, it starts to monitor its ports. For each port, `ttymon` first initializes the STREAMS modules for the line disciplines, if they are specified, and the speed and terminal settings. The values used for initialization are taken from the appropriate entry in the TTY settings file. This file is maintained by the `sttydefs(1M)` command. Default STREAMS modules for ports are usually set up by the `autopush` command of the Autopush Facility.

`ttymon` then writes the prompt and waits for user input. If the user indicates that the speed is inappropriate by pressing the BREAK key, `ttymon` tries the next speed and writes the prompt again. When valid input is received, `ttymon` interprets the per-service configuration file for the port, if one exists, creates a `utmp` entry if required, establishes the service environment, and then invokes the service associated with the port. Valid input consists of a string of at least one non-newline character, terminated by a carriage return. After the service terminates, `ttymon` cleans up the `utmp` entry, if one exists, and returns the port to its initial state.

If the *pre-prompt-config* flag is set for a connection, the configuration file is interpreted before the prompt is output and is not interpreted before the service starts.

If *autobaud* is enabled for a port, `ttymon` will try to determine the baud rate on the port automatically. Users must enter a carriage return before `ttymon` can recognize the baud rate and print the prompt. Currently, the baud rates that can be determined by *autobaud* are `110`, `1200`, `2400`, `4800`, and `9600`.

If a port is configured as a bidirectional port, `ttymon` will allow users to connect to a service, and, if the port is free, will allow `uucico`, `cu` or `ct` to use it for dialing out. If a port is bidirectional, `ttymon` will wait to read a character before it prints a prompt.

If the *connect-on-carrier* flag is set for a port, `ttymon` will immediately invoke the port's associated service when a connection request is received. The prompt message will not be sent.

If a port is disabled, `ttymon` will not start any service on that port. If a disabled message is specified, `ttymon` will send out the disabled message when a connection request is received. If `ttymon` is disabled, all ports under that instance of `ttymon` will also be disabled.

Service invocation

The service `ttymon` invokes for a port is specified in the `ttymon` administrative file. `ttymon` will scan the character string giving the service to be invoked for this port, looking for a `%d` or a `%%` two-character sequence. If `%d` is found, `ttymon` will modify the service command to be executed by replacing those two characters by the full path name of this port (the device name). If `%%` is found, they will be replaced by a single `%`.

When the service is invoked, the file descriptors `0`, `1`, and `2` are opened to the port device for reading and writing. The service is invoked with the user ID, group ID and current `HOME` directory set to that of the user name under which the service was registered with `ttymon`. Two environment variables, `HOME` and

TTYPROMPT, are added to the service's environment by ttymon. HOME is set to the HOME directory of the user name under which the service is invoked. TTYPROMPT is set to the prompt string configured for the service on the port. This is provided so that a service invoked by ttymon has a means of determining if a prompt was actually issued by ttymon and, if so, what that prompt actually was.

If NONE is specified for the service, no service is started. ttymon only keeps the interface open in this case until a SIGHUP occurs on the interface. Then the connection is reset to the initial state.

See ttyadm(1M) for options that can be set for ports monitored by ttymon under the Service Access Controller.

Repeated service start

Three variables in /etc/default/ttymon can influence the repeated start of services or the initialization of connections. If SPAWN_LIMIT is positive and a SPAWM_LIMIT service often has to be initialized within SPAWN_INTERVAL seconds, it is suspended for SPAWN_INHIBIT seconds before another attempt is made to initialize the connection. If SPAWN_INHIBIT = 0, the service is suspended until someone enters the command sacadm –p *pmtag* –x.

Environment variables

ttymon can be configured using environment variables. The variables are set preferably in the monitor-specific configuration file /etc/saf/*pmtag*/_config [see doconfig(3C)]. The following variables are recognized:

TM_SERV_REPORT
: If "off" is set, starting a service is not logged in the log file. The default is "on".

TM_OPEN_TIME
: Time in milliseconds that the initialization of a connection may require. If the initialization cannot be completed in this time, the connection is closed and another attempt may be made at a later stage. If TM_OPEN_TIME is not set, TM_OPEN_TIME = 1900 is set.

TM_DELAY_TIME
: Time in seconds that a service is suspended if an error occurred on the relevant connection before another attempt could be made to initialize the connection. The value must not be negative. If this is not the case or TM_DELAY_TIME is not set, TM_DELAY_TIME = 5 is set. Please note that the repeated start of services in the case of errors is influenced by this variable, see section above.

Invoking a stand-alone ttymon process

A special invocation of ttymon is provided with the –g option. This form of the command should only be called by applications that need to set the correct baud rate and terminal settings on a port and then connect to login service, but that cannot be pre-configured under the SAC. The following combinations of options can be used with –g:

-d *device*
: *device* is the full path name of the port to which ttymon is to attach. If this option is not specified, file descriptor 0 must be set up by the invoking process to a TTY port.

–h
: If the –h flag is not set, ttymon will force a hangup on the line by setting the speed to zero before setting the speed to the default or specified speed.

–t *timeout*
: Specifies that ttymon should exit if no one types anything in *timeout* seconds after the prompt is sent.

–l *ttylabel*
: *ttylabel* is a link to a speed and TTY definition in the ttydefs file. This definition tells ttymon at what speed to run initially, what the initial TTY settings are, and what speed to try next if the user indicates that the speed is inappropriate by pressing the BREAK key. The default speed is 9600 baud.

–p *prompt*
: Allows the user to specify a prompt string. The default prompt is "Login: ".

–m *modules*

When initializing the port, `ttymon` will pop all modules on the port, and then push *modules* in the order specified. *modules* is a comma-separated list of pushable modules. Default modules on the ports are usually set up by the Autopush Facility.

NOTES

If a port is monitored by more than one `ttymon`, it is possible for the `ttymon`s to send out prompt messages in such a way that they compete for input.

If variables are being set for the *pre-prompt-config* flag in the monitor-specific configuration file, then this should take place before the first `run` or `runwait` command.

SEE ALSO

`pmadm`(1M), `sac`(1M), `sacadm`(1M), `sttydefs`(1M), `ttyadm`(1M).

System Administrator's Reference Manual

tunefs(1M)

NAME

`tunefs -` tune up an existing UFS file system

SYNOPSIS

`tunefs` [–a *maxcontig*] [–d *rotdelay*] [–e *maxbpg*] [–m *minfree*] [`-o` {`space` | `time`}] *special* | *filesystem*

DESCRIPTION

`tunefs` is designed to change the dynamic parameters of a file system, which determine the layout of a file system. The file system must be unmounted beforehand with `tunefs`. The parameters to be modified are indicated by the options given below.

When a `ufs` file system is created with `newfs`, these parameters can be set as required. These parameters should only be modified for special applications and can only be modified by system administrators, who have very good knowledge of how `ufs` file systems are organized.

OPTIONS

–a *maxcontig*

Specifies the maximum number of contiguous blocks that can be transferred before the disk access is interrupted. The default value is one, since most device drivers require one interrupt per disk transfer. For device drivers that can chain several blocks together in a single transfer, you should enter the maximum length for this type of chain. In the case of RM400, RM600-xx, and RM600 systems, this value is always 1.

–d *rotdelay*

Specifies the expected time (in milliseconds) to complete a transfer and initiate a new transfer on the same disk. This value is used to decide how much rotational spacing to place between successive blocks in a file.

–e *maxbpg*

Indicates the maximum number of blocks any single file can allocate out of a cylinder group before it is forced to begin allocating blocks from another cylinder group. Typically, the value for *maxbpg* is approximately 25% of the total blocks in a cylinder group.

This option is used to prevent any single file from using up all the blocks in a single cylinder group, thus degrading access times for all files subsequently allocated in that cylinder group.

The effect of this limit is to cause big files to do long seeks more frequently than if they were allowed to allocate all the blocks in a cylinder group before seeking elsewhere. For file systems with exclusively large files, this parameter should be set higher.

–m *minfree*

Specifies the percentage of space retained for normal users. As soon as the capacity of the file system reaches this threshold, only an authorized user can allocate space for files. The default value is 10%.

If the specified value exceeds the current usage level, users will be unable to allocate files until enough files have been deleted to get under the higher threshold.

The threshold can also be set to zero, however up to a factor of three in throughput will be lost compared to the performance obtained at 10%.

–o {`space` | `time`}

Changes optimization strategy for the file system. (The optimization strategy can be defined when the file system is created. If this is not defined, the default `time` applies.).

`space`
Conserves space.

`time` Minimizes access time

You should generally optimize for shorter access times, unless the file system is over 90% full.

SEE ALSO

`mkfs`(1M), `newfs`(1M), `fork`(2), `terminfo`(4).

System Administrator's Reference Manual

tunestat(1M)

NAME

`tunestat` - check resource usage of tunable parameters

SYNOPSIS

`/usr/sbin/tunestat` [–n *namelist*] [–p *percent*] [–q] [*variable* ...]

DESCRIPTION

The `tunestat` command prints the resource usage of structures that are allocated based upon tunable parameters in `mtune(4)` and `stune(4)`. `tunestat` is actually a shell script that uses `crash(1M)` and `ikdb(8)` to retrieve most of the information.

The `tunestat` command will print out the variable that tracks the structure (the name is in the form of `tune_`*structure-name*), the current usage, the high water mark since the kernel was booted, the maximum allowed by `mtune(4)` or `stune(4)`, the percentage of the high water mark versus the maximum, and the associated tunable parameter(s) that can be modified. If no variables are given, all the structures that the `tunestat` command knows about will be printed. If a variable is not a valid name, an error message will be printed and the command will continue with the rest of the variables.

Command line arguments to `tunestat` are *namelist*, *percent*, and *variable*.

The text file *namelist* contains the symbol table information needed for symbolic access to the system memory image to be examined. The default *namelist* is `/stand/unix`.

percent is used to print out only those structures whose peak usage compared to the maximum is greater than or equal to *percent*.

The `–q` option invokes the quiet mode; the table header and errors will not be printed.

EXAMPLES

The command to print out the statistics for `tune_proc`, the number of number of processes in the system, is:

```
tunestat tune_proc
```

The output would be:

```
Variable     Current   Peak   Maximum   Peak %   Tunable Parameters
--------     -------   ----   -------   ------   ------------------
tune_proc         36     40      4116        0   "NPROC"
```

DIAGNOSTICS

`invalid variable:`

The argument *variable* given to `tunestat` is not in the kernel. Using `tunestat` without any variable arguments will print all the variables known to the kernel.

`<percent> must be between 0 and 100`

When using the `–p` option, *percent* must be between 0 and 100.

`tunestat: cannot start crash`

For some reason, `tunestat` was unable to start `crash(1M)`. A common problem is that either the default namelist or the argument given to the `–n` option is incorrect.

FILES

`/dev/mem`
: system image of currently running system

SEE ALSO

`crash(1M)`, `mtune(4)`, `stune(4)`, `ikdb(8)`.

uadmin(1M) - zic(1M)

uadmin(1M)

NAME

`uadmin` - administrative control

SYNOPSIS

`/sbin/uadmin` *cmd fcn*

DESCRIPTION

The `uadmin` command provides control for basic administrative functions. This command is tightly coupled to the system administration procedures and is not intended for general use. It may be invoked only by the superuser.

The arguments *cmd* (command) and *fcn* (function) are converted to integers and passed to the `uadmin` system call.

SEE ALSO

`uadmin`(2) .

System Administrator's Reference Manual

ufsdump(1M)

NAME

`ufsdump` - incremental file system dump

SYNOPSIS

`/sbin/ufsdump` [*options*] *filesystem*

DESCRIPTION

`ufsdump` back up all files in *filesystem*, or files changed after a certain date, to an output device or file; *options* is a string that specifies `ufsdump` options, as shown below; *filesystem* is a character special device (for example, `/dev/ios0/rsdisk000s4`) or a mount point for a file system (for example, `/usr`).

If no *options* are given, the default is `9u` and the default output device is assumed to be a 2400 foot, 6250 bpi 9-track tape.

If the `c` option is specified, the output device is assumed to be a QIC-320 cartridge tape.

If the `D` option is specified, the output device is assumed to be a DAT (Digital Audio Tape) device.

Options `c`, `d`, `D`, and `s` are only used to calculate tape size. The `M` option is used to specify megabytes per tape, and may be used with any type of tape. The `M` option is much simpler than the above options, provided the tape capacity is known.

`ufsdump` does not handle EOT (end of tape) gracefully. Unlike other backup utilities, such as `cpio`, that detect the EOT and then prompt the user to insert another tape before continuing the backup, `ufsdump` will issue a write error upon reaching the EOT and will attempt to restart the backup. It is, therefore, important to supply `ufsdump` with accurate information about the tape capacity with the `M` option, or about tape type, density, and length. This will allow `ufsdump` to determine how much data to write to each tape before asking for the next tape. (See **EXAMPLES** below.) Since `ufsdump` doesn't handle EOT, it is important to be aware that if you over-estimate how much data fits on the tape, the last of the multiple dumps on the tape will be incomplete. You can store multiple dumps on one tape by writing to the no-rewind device, although you may need to restart the last dump when it hits EOT.

Example:

```
ufsdump 0uf /dev/ios0/rstape000hn /usr
ufsdump 0uf /dev/ios0/rstape000h /
```

The above commands write a level `0` dump of `/usr` and a level `0` dump of `/` (root) to the same tape. Notice the use of the no-rewind device in the dump of `/usr`. Multiple dumps can be recovered using the `s` option in `ufsrestore`(1M) .

Use the following guidelines, keeping in mind that file systems with many small files (<2 KB) will use more tape, and that use of the `M` option is recommended in lieu of other options which determine tape size (`c`, `d`, `D`, and `s`):

Tape	Mode	Type	Length	Capacity
9-track	–	2400	6250ft	~140 MB
9-track	–	2400	1600ft	~25 MB
Streamer:				
QIC-320	–	600	–	~120 MB
QIC-120	–	120M	620ft	~105 MB
QIC-150	–	150M	620ft	~135 MB
QIC-525	–	525M	620ft	~475 MB

DAT/DDS-1	–	–	60m	~1200 MB
DAT/DDS-1	–	–	90m	~1800 MB
DAT/DDS-2	–	–	120m	~3600 MB
8mm Tape	EXB-8200	–	15m	~265 MB
(Exabyte)	EXB-8500	–	15m	~530 MB

Notes:

- EXB-8200 and EXB-8500 are different write densities for working with 8 mm tapes (low- and high-density).
- If a "device with compression" is being used (e.g. `/dev/ios0/rstape004c`), this increases the capacity by a factor of 1.6 – 2.0 (maximum).
- The capacity of all "tapes" is approx. 10% below the specified sizes (e.g. 150 MB streamer can record approx. 135 MB data). The reduction in the specified value takes into account any tape errors and automatic tape leader for lower transfer rates.
- Specific details on all tape types (e.g. how a 150 MB streamer behaves in a 525 MB streamer drive) can be found in the **TAPE NOTES** for `stape(7)`.

OPTIONS

`0–9` Dump level. All files in the *filesystem* that have been modified since the last `ufsdump` at a lower dump level are copied to the volume. For instance, if you did a level `2` dump on Monday, followed by a level `4` dump on Tuesday, a subsequent level `3` dump on Wednesday would contain all files modified or added since the level `2` (Monday) backup. A level `0` dump copies the entire *filesystem* to the dump volume.

`b` *factor*
Blocking factor. Specifies the blocking factor for tape writes. The default is `64` blocks per write. A tape block is 512 bytes. Note that the blocking factor must be a multiple of 4.

`c` Cartridge. Uses a QIC-320 instead of the standard half-inch reel. The length is set to 560 feet. This option is incompatible with the `d` and `s` options. If the tape capacity is known, the `M` option is recommended instead of the `c` option.

`d` *bpi* Tape density. Meaningful only for 9-track tapes. The density of the tape, expressed in bits per inch, is taken from *bpi*. This is used to keep a running tab on the amount of tape used per reel. The default density is `6250`.

`D` *mins*
DAT. Uses DAT (Digital Audio Tape) instead of the standard half-inch reel. *mins* is the number of minutes on the DAT tape. This option is incompatible with the `d` and `s` options.

`f` *dump-file*
Dump file. Uses *dump-file* as the output device or file to dump to, instead of the default (`/dev/rmt8`). If *dump-file* is specified as –, dump to the standard output. Note that *dump-file* can be the name of a disk file (for example, `/tmp/dump.output`). This creates a dump image on disk. This can then be restored (for example, `ufsrestore xuf /tmp/dump.output`), and transferred to another system or written to tape using another utility, such as `cpio`.

`M` *megabytes*
Maximum amount of data to write to each tape, given in megabytes (10^6). This option determines the maximum amount of data to write to each tape before `ufsdump` will ask for another tape. This can be used with QIC, DAT, 9-track, or any type of tape device. The maximum *megabytes* specified should be somewhat less than the capacity of the tape, to allow for retries on bad sections of tape. This option overrides the `c`, `d`, `D`, and `s` options.

`n` Notifies all operators in the operator group that `ufsdump` requires attention by sending messages to

their terminals, in a manner similar to that used by the `wall` command.

`s` *size*
: Specifies the *size* (in feet) of the volume being dumped to. When the specified *size* is reached, `ufsdump` waits for you to change the volume. The default *size* for 9-track tape is `2300` (feet), which is slightly less than the actual physical size of the tape. This allows for bad sectors on the tape and guards against tape overrun.

`u`
: Updates the dump record. For each file system successfully dumped, adds an entry to the file `/etc/dumpdates`, that includes the file system name, date, and dump level. This file can be edited by the superuser.

`W`
: The `W` option lists all file systems that appear in `/etc/dumpdates` and `/etc/fstab`, along with information about their most recent dump dates and levels. If the `W` option is set, all other options are ignored, and `ufsdump` exits immediately.

EXAMPLES

The following examples describe possible command lines using `ufsdump` options:

`ufsdump 0Muf 320 /dev/ios0/rstape004 /usr`
: Perform a level `0` dump of `/usr` to `/dev/ios0/rstape004`, which writes a maximum of 320 MB to each tape before asking for the next tape. The type of tape device does not matter, but you must know the tape capacity.

`ufsdump 0udf 1600 /dev/ios0/rstape004 /usr`
: Perform a level `0` dump of `/usr` to `/dev/ios0/rstape004`, which is a 1600 bpi 9-track tape drive with a 2400 foot tape mounted on it; update `/etc/dumpdates`.

`ufsdump 0usf 1200 /dev/ios0/rstape001h /usr`
: Perform a level `0` dump of `/usr` to `/dev/ios0/rstape001h`, which is a 6250 bpi 9-track tape drive with a 1200 foot tape mounted on it; update `/etc/dumpdates`.

`ufsdump 2ucf /dev/ios0/rstape000 /home`
: Perform a level `2` dump of `/home` to `/dev/ios0/rstape000`, which is a QIC-320 cartridge tape; update `/etc/dumpdates`.

`ufsdump 0Df 120 /dev/ios0/rstape000 /usr`
: Perform a level `0` dump of `/usr` to `/dev/ios0/rstape000`, which is a DAT 60-meter (120 minute) tape; do not update `/etc/dumpdates`.

NOTES

Fewer than 32 read errors on the file system are ignored.

Each tape requires a new process, so parent processes for tapes already written just hang around until the entire archive is written.

It is recommended that incremental dumps also be performed with the system running in singleuser mode.

For a multiple-tape dump. Always use tapes of exactly the same capacity, to avoid hitting EOT on a shorter tape.

For simplicity, if the tape size in megabytes is known, the `M` option is recommended, instead of the `c`, `d`, `D`, or `s` options.

If you do multiple dumps to a "no-rewind" tape device file, the maximum capacity to write to each tape will not protect `ufsdump` from eventually hitting EOT. This may require that you restart the last (failed) dump on a new tape.

SECURITY NOTE

Only a superuser can do a dump of a file system. This prevents users from being able to gain access to files with selective read permissions.

NOTES

The `s` option should not be used with the `c` option. If these options are used together, the number of feet

specified will be multiplied by 4 resulting in erroneous capacities per tape.

`ufsdump` does not gracefully handle EOT.

FILES

`/etc/dumpdates`
: dump date record

`/etc/group`
: to find group operator

SEE ALSO

`tar`(1), `shutdown`(1M), `ufsrestore`(1M), `wall`(1M), `stape`(7).

System Administrator's Reference Manual

ufsrestore(1M)

NAME

`ufsrestore` - incremental file system restore

SYNOPSIS

`/sbin/ufsrestore` *options* [*filename ...*]

DESCRIPTION

`ufsrestore` restores files from backup tapes created with the `ufsdump(1M)` command. *options* is a string of at least one of the options listed below, along with any modifiers and arguments you supply. Remaining arguments to `ufsrestore` are the names of files (or directories) which are to be restored to disk. Unless the `h` modifier is in effect, a directory name refers to the files it contains, and (recursively) its subdirectories and the files they contain.

OPTIONS

`i` Interactive. After reading in the directory information from the tape, `ufsrestore` invokes an interactive interface that allows you to browse through the dump tape's directory hierarchy and select individual files to be extracted. See **INTERACTIVE COMMANDS**, below, for a description of available commands.

`r` Restores the entire tape and loads the tape's full contents into the current directory. This option should be used only to restore a complete dump tape onto a clear filesystem, or to restore an incremental dump tape after a full "level `0`" restore.

`R` Resumes restoring. `ufsrestore` requests a particular tape of a multivolume set from which to resume a full restore (see the `r` option above). This allows `ufsrestore` to start from a checkpoint when it is interrupted in the middle of a full restore.

`t` Table of contents. Lists each *filename* that appears on the tape. If no *filename* argument is given, the root directory is listed. This results in a list of all files on the tape, unless the `h` modifier is in effect.

`x` Extracts the named files from the tape. If a named file matches a directory whose contents were written onto the tape, and the `h` modifier is not in effect, the directory is recursively extracted. The owner, modification time, and mode are restored (if possible). If no *filename* argument is given, the root directory is extracted. This results in the entire tape being extracted unless the `h` modifier is in effect.

`c` Converts the contents of the dump tape to the new filesystem format.

`d` Debug. Turns on debugging output.

`h` Extracts the actual directory, rather than the files that it references. This prevents hierarchical restoration of complete subtrees from the tape.

`m` Extracts by inode numbers rather than by filename to avoid regenerating complete pathnames. This is useful if only a few files are being extracted.

`v` Verbose. `ufsrestore` displays the name of each file it restores, preceded by its file type.

`y` Does not ask whether to abort the restore in the event of tape errors. `ufsrestore` tries to skip over the bad tape block(s) and continue as best it can.

`b` *factor*
Blocking factor. Specifies the blocking factor for tape reads.

Note: A tape block is 512 bytes.

`f` *dump-file*
Uses *dump-file* instead of `/dev/rmt8` as the file to restore from. If *dump-file* is specified as "–", `ufsrestore` reads from standard input. This allows, `ufsdump(1M)` and `ufsrestore` to be used in a pipeline to dump and restore a file system:

```
ufsdump 0f – /dev/ios0/rsdisk000s3 | (cd /mnt; ufsrestore xf –)
```

`s` *n* Skips to the *n*'th file when there are multiple dump files on the same tape. For example, the command:

```
ufsrestore xfs /dev/ios0/rstape000h 5
```

would position you at the fifth file on the tape.

INTERACTIVE COMMANDS

`ufsrestore` enters interactive mode when invoked with the `i` option. Interactive commands are reminiscent of the shell. For those commands that accept an argument, the default is the current directory.

`ls` [*directory*]
: List files in *directory* or the current directory, represented by a "`.`" (period). Directories are appended with a "`/`" (backslash). Entries marked for extraction are prefixed with a "`*`" (asterisk). If the verbose option is in effect, inode numbers are also listed.

`cd` *directory*
: Change to directory *directory* (within the dump-tape).

`pwd` Print the full pathname of the current working directory.

`add` [*filename*]
: Add the current directory, or the named file or directory *directory* to the list of files to extract. If a directory is specified, add that directory and its files (recursively) to the extraction list (unless the `h` modifier is in effect).

`delete` [*filename*]
: Delete the current directory, or the named file or directory from the list of files to extract. If a directory is specified, delete that directory and all its descendents from the extraction list (unless the `h` modifier is in effect). The most expedient way to extract a majority of files from a directory is to add that directory to the extraction list, and then delete specific files to omit.

`extract`
: Extract all files on the extraction list from the dump tape. `ufsrestore` asks which volume the user wishes to mount. The fastest way to extract a small number of files is to start with the last tape volume and work toward the first. The owner, modification time, and mode are restored (if possible).

`verbose`
: Toggle the status of the `v` modifier. While `v` is in effect, the `ls` command lists the inode numbers of all entries, and `ufsrestore` displays information about each file as it is extracted.

`help` Display a summary of the available commands.

`quit` `ufsrestore` exits immediately, even if the extraction list is not empty.

SECURITY NOTES

Only a superuser can do a dump of a filesystem. This prevents users from being able to gain access to files with selective read permissions.

There are special considerations for specifying the remote username in the *dump-file* to the `f` option. In order to specify a user other than the one who invoked the remote dump or restore utility, the user who invoked the utility must have permission from the remote user. This is accomplished by making the appropriate entry in the `.rhosts` file which resides in the remote user's home directory. The entry should include the originating hostname and username (which will always be "root" since only a superuser can dump a filesystem).

In addition to the normal security checks, the security parameters can affect the use of the remote dump and restore utilities.

If the `AUTO_LOGIN_MIN_UID` parameter is set to a positive number on the remote host system, then users with user ids less than that number cannot use automatic remote access.

If the `DISABLE_RHOSTS` parameter is set to "yes" on the remote host system, then any `.rhosts` files will be ignored on the remote host system. If the `DISABLE_RHOSTS` parameter is set to "no" on the remote host system, then any `.rhosts` files must be writable only by the owner.

NOTES

`ufsrestore` can get confused when doing incremental restores from dump tapes that were made on active file systems.

A "level `0`" dump must be done after a full restore. Because `ufsrestore` runs in user mode, it has no control over inode allocation; this means that `ufsrestore` repositions the files, although it does not change their contents. Thus, a full dump must be done to get a new set of directories reflecting the new file positions, so that later incremental dumps will be correct.

DIAGNOSTICS

`ufsrestore` complains about bad option characters.

Read errors result in complaints. If `y` has been specified, or the user responds `y`, `ufsrestore` will attempt to continue.

If the dump extends over more than one tape, `ufsrestore` asks the user to change tapes. If the `x` or `i` option has been specified, `ufsrestore` also asks which volume the user wishes to mount.

There are numerous consistency checks that can be listed by `ufsrestore`. Most checks are self-explanatory. Common errors are given below.

`Converting to new file system format.`
: A dump tape created from the old file system has been loaded. It is automatically converted to the new file system format.

filename`: not found on tape`
: The specified file name was listed in the tape directory, but was not found on the tape. This is caused by tape read errors while looking for the file, and from using a dump tape created on an active file system.

`expected next file` *inumber*`, got` *inumber*
: A file that was not listed in the directory showed up. This can occur when using a dump tape created on an active file system.

`Incremental tape too low`
: When doing an incremental restore, a tape that was written before the previous incremental tape, or that has too low an incremental level has been loaded.

`Incremental tape too high`
: When doing incremental restore, a tape that does not begin its coverage where the previous incremental tape left off, or one that has too high an incremental level has been loaded.

`Tape read error while restoring` *filename*
`Tape read error while skipping over inode` *inumber*
`Tape read error while trying to resynchronize`
`A tape read error has occurred.`
: If a file name is specified, then its contents are probably partially wrong. If an inode is being skipped or the tape is trying to resynchronize, then no extracted files have been corrupted, though files may not be found on the tape.

`resync ufsrestore, skipped` *num*
: After a tape read error, `ufsrestore` may have to resynchronize itself. This message lists the number of blocks that were skipped over.

FILES

`/dev/rmt8`
: the default tape drive

`/tmp/rstdir*`
: file containing directories on the tape

`/tmp/rstmode*`
: owner, mode, and timestamps for directories

`./restoresymtable`
: information passed between incremental restores

SEE ALSO

`ufsdump`(1M), `mkfs`(1M), `mount`(1M).

System Administrator's Reference Manual

ups(1M)
RM600 only

NAME

ups - command for starting the ubft daemon and communicatingwith the EIP

SYNOPSIS

ups [–D] [–l *size*] –d

ups –s [–v]

ups –v

ups *eipcmd*

DESCRIPTION

The ubft daemon monitors the power supply, temperature of the air circulation, function of the fans and other hardware components on the system. Furthermore, the ubft daemon monitors most of the emergency power supply options for the system. These include the external BBU cabinets (Battery Backup Unit), internal BBUs, integrated in the cabinets, and power packs combined with a BBU. The uninterruptible power supply, also known as UPS, is also included here.

Any errors detected are reported on the console terminal and in the system log by the ubft daemon; the error messages contain the string ubft in the ID. The ubft daemon also has a private ubft logging file.

When the system is switched to administrator or multiuser mode, the ubft daemon is started using an rc script, and the –d option is used in this case.

OPTIONS

–d By specifying this option, a ubft daemon is started. If another ubft daemon is already running, the program terminates with a corresponding error message.

–l *size*

This option defines the maximum size of the ubft logging file /var/adm/ubft/log with the value *size* in bytes. If the value 0 is specified, logging is deactivated in the file. A value other than 0 activates the logging. If a value greater than 0 and less than 1024 is specified for *size*, the size of the ubft logging file is restricted to 1024 bytes. If the –l option is not specified, 1048676 (1 MB) is taken as the default and logging is switched on.

If the specified size of the file is exceeded while the system is running, the old file is renamed to /var/adm/ubft/log._old and logging continues in an empty file /var/adm/ubft/log.

–s [–v]

The –s option is used to determine the current system status with regard to the components being monitored by the ubft daemon. Furthermore, the version of the ubft daemon, the versions of the CSI/HIOS/EHIOS and EIP firmware, and other system and configuration parameters are output to standard output.

By additionally specifying the –v option, all available components are displayed for every cabinet and not just the faulty ones. The –v option only works on systems with the EIP3 master. It is ignored on systems with the EIP2 master.

–D The –D option switches on a trace which is stored in the /var/adm/ubft/trace file. This file has a maximum size of 1 MB and if it exceeds this size, it is renamed to /var/adm/ubft/trace._old and the trace continues in an empty /var/adm/ubft/trace file.

–v Outputs the version of the ubft daemon to stdout.

If no option is specified, the subsequent parameters are interpreted as an *eipcmd* EIP command. The EIP command is sent to the EIP and the EIP response is sent to standard output. Example: /sbin/ups ir 0 70 returns the EIP response BF>. If no parameters are specified, EIP commands are expected by standard

input.

NOTES

The `/sbin/ups`*eipcmd* command initiates the `/etc/opt/havail/eip2cmd`*eipcmd* command and should be used in its place.

The maximum amount of data that can be requested using an *eipcmd* EIP command depends on the central boot controller in the system cabinet: In systems with HIOS/EHIOS masters, the maximum amount of data is 4 KB, in systems with CSI masters, the maximum is 2 KB.

The `ups` command determines itself whether an EIP2 or an EIP3 is used in the system and, if necessary, calls the `/sbin/ups2` program or `/sbin/ups3` program with appropriate parameters. This means that the `ubft` daemon process is called "ups2" or "ups3".

NOTES

The `ups` command can only be executed by system administrators.

The response times and behavior of the `ubft` daemon can be assigned parameters in the `/var/adm/ubft/_upstab` file. This file is evaluated when the `ubft` daemon is started. While it is running, the `ubft` daemon can be made to evaluate the parameter file once again by issuing the `kill -2` *ups_pid* command. The process number of the `ubft` daemon must be specified in this call for *ups_pid*.

The `ubft` daemon can only be terminated using `kill -9`*ups_pid*.

In the event of serious errors, the daemon terminates with a return value greater than 0.

FILES

`/dev/ubft`
: Special file of the `ubft` driver used by the `ubft` daemon.

`/var/adm/ubft/log`
: File where the `ubft` daemon stores its `ubft` logging messages.

`/var/adm/ubft/trace`
: File containing the trace.

`/var/adm/ubft/state`
: File containing internal structures of the `ubft` daemon.

`/var/adm/ubft/_upstab`
: File used to assign parameters to the `ubft` daemon.

SEE ALSO

`_upstab`(4), `ubft`(7), `user_reaction`(7).

System Administrator's Reference Manual

ups(1M)
RM400 only

NAME

ups - uninterruptible power supplycontrol

SYNOPSIS

ups [–d]

DESCRIPTION

If a system is backed up with UPS (Uninterruptible Power Supply), the system behavior in the event of a power failure can be defined using the ups command (called without any parameters). Parameters are entered interactively:

1. "Time from line fail until start of shutdown procedure (in seconds)"

   If power is lost and is not restored within a specific time frame, the system is shut down.

   Default time frame: 20 seconds

2. "Time to wait until battery is loaded (in minutes)"

   Following a power failure, the accumulator of a battery can be fully discharged. To recharge the battery, a specified period of time should elapse in "safe boot" mode (see point 4) after power has been restored.

   Default time frame: 120 minutes

3. "Shutdown behavior (0 = normal shutdown, 1 = immediate shutdown)"

   In a "normal shutdown", the user applications entered in the shell procedure /etc/usr.shutdown are terminated when this procedure is started. The reason for the shutdown is then sought. If none can be found, the operating system is not terminated. Otherwise, the computer operation is terminated correctly with the shell script /sbin/shutdown. If "immediate shutdown" is selected, the computer operation is terminated by calling the /sbin/uadmin 2 0 command.

   Default setting: 0 = normal shutdown

4. "Boot behavior (0 = safe boot, 1 = temporary unsafe boot)"

   This setting defines the operating mode for the boot process following a power failure. If 0 is entered, a "safe boot" occurs, i.e. after power has returned and the system is brought up, there is a pause until the batteries have been properly loaded. This time period is specified under point 2 (see above). Multiuser mode is only permitted after this time has elapsed.

   If 1 is specified, a "temporary unsafe boot" occurs, i.e. the system starts multiuser mode immediately.

   Default setting: 1 = temporary unsafe boot

5. "Quit"

   The ups command is terminated and the values set are activated.

The ups command with the –d option activated the ups daemon. If there is a problem with the power supply, the ups daemon initiates a shutdown. The ups daemon is started when Reliant UNIX is being brought up.

NOTES

The ups command can only be executed by the system administrator.

A system can only be backed up using either BBU or UPS. Simultaneous operation is not supported.

The sum of the daemon waiting time and the process runtime must not exceed the maximum battery bridging time. It is also worth considering having a reserve for a further power failure.

The maximum stored energy time in the batteries for the UPS depends on the type being used (please refer to the documentation supplied with UPS).

System Administrator's Reference Manual

useradd(1M)

NAME

useradd - administer a new user login on the system

SYNOPSIS

useradd [–u *uid* [–o]] [–g *group*] [–G *group*[, *group* ...]] [–d *dir*] [–s *shell*] [–c *comment*] [–m [–k *skel_dir*]] [–f *inactive*] [–e *expire*] [–a *event*] *login*

useradd –D [–g *group*] [–b *base_dir*] [–f *inactive*] [–e *expire*]

DESCRIPTION

Invoking useradd without the –D option adds a new user entry to the /etc/passwd and /etc/shadow files. It also creates supplementary group memberships for the user (–G option) and creates the home directory (–m option) for the user if requested. The new login remains locked until the passwd(1) command is executed.

Invoking useradd –D with no additional options displays the default values for *group*, *base_dir*, *shel_dir*, *shell*, *inactive*, and *expire*. The values for *group*, *base_dir*, *inactive*, *expire*, and *shell* are used for invocations without the –D option.

Invoking useradd –D with –g, –b, –f, or –e (or any combination of these) sets the default values for the respective fields. [As installed, the default group is other (group ID of 1) and the default value of *base_dir* is /home.] Subsequent invocations of useradd without the –D option use these arguments.

The system file entries created with this command have a limit of 512 characters per line. Specifying long arguments to several options may exceed this limit.

OPTIONS

–a *event*[,...]
: A comma-separated list of event types or classes that make up the user's audit mask. There is no default user audit mask in the system as delivered, but you can define a default user audit mask in the file /etc/default/useradd using the defadm command. This option is only valid if the auditing subsystem is installed.

–b *base_dir*
: The default base directory for the system. If –d *dir* is not specified. *base_dir* is concatenated with the user's login to define the home directory. If the –m option is not used, *base_dir* must exist.

–c *comment*
: Any text string. It is generally a short description of the login, and is currently used as the field for the user's full name. This information is stored in the user's /etc/passwd entry.

–d *dir*
: The home directory of the new user. It defaults to *base_dir*/*login*, where *base_dir* is the base directory for new login home directories and *login* is the new login.

–e *expire*
: The date on which a login can no longer be used; after this date, no user will be able to access this login. (This option is useful for creating temporary logins.) You may type the value of the argument *expire* (which is a date) in various formats (included in /etc/datemsk). For example, you may enter 10/6/01 or October 6, 2001.

  Note:

  The date is indicated in Greenwich Mean Time (GMT) and may therefore differ by as much as a day in other time zones. For example, if an expiry date only comes into effect in Central European Time CET at 1 am and not at midnight, which would correspond to GMT.

–f *inactive*
: The maximum number of days allowed between uses of a login ID before that login ID is declared

invalid. Normal values are positive integers. A value of `0` defeats the status.

`-G` *group*
An existing group's integer ID or character-string name. It defines the new user's supplementary group membership. Duplicates between *group* with the `-g` and `-G` options are ignored. No more than `NGROUPS_MAX` groups may be specified.

`-g` *group*
An existing group's integer ID or character-string name. Without the `-D` option, it defines the new user's primary group membership and defaults to the default group. You can reset this default value by invoking `useradd -D -g` *group*.

`-k` *skel_dir*
A directory that contains skeleton information (such as `.profile`) that can be copied into a new user's home directory. This directory must exist. The system provides a "skel" directory (`/etc/skel`) that can be used for this purpose.

`-m` Creates the new user's home directory if it doesn't already exist. If the directory already exists, it must have read, write, and execute permissions for *group*, where *group* is the user's primary group. If the access permissions do not match, `useradd` outputs a warning, but the command does not terminate.

`-o` This option allows a UID to be duplicated (non-unique).

`-s` *shell*
Full pathname of the program used as the user's shell on login. It defaults to an empty field causing the system to use `/sbin/sh` as the default. The value of *shell* must be a valid executable file.

`-u` *uid*
The UID of the new user. This UID must be a non-negative decimal integer which is less than `MAXUID` as defined in `<param.h>`. The UID defaults to the first available free UID. For example, if UIDs 100, 105, and 200 are assigned, the next default UID will be 101. (UIDs from 0–99 are reserved.)

Note:

This assignment method only works for *uid* < 3000. If *uid uid* ≥ 3000, you will need the `SIfmllan` package.

login
A string of printable characters that specifies the new login name of the user. It may not contain a colon (`:`) or a newline (`\n`).

DIAGNOSTICS

The `useradd` command exits with one of the following values:

`0` = The command was executed successfully.

`2` = The command line syntax was invalid. A usage message for the `useradd` command is displayed.

`3` = An invalid argument was provided with an option.

`4` = The *uid* specified with the `-u` option is already in use.

`6` = The *group* specified with the `-g` option does not exist.

`9` = The specified *login* is not unique.

`10` = Cannot update `/etc/group`. The login was added to the `/etc/passwd` file but not to the `/etc/group` file.

`12` = Unable to create the home directory (with the `-m` option) or unable to complete the copy of *skel_dir* to the home directory.

LOCALE

The `LC_MESSAGES` environment variable governs the language in which message texts are displayed.

If `LC_MESSAGES` is undefined or is defined as the null string, it defaults to the value of `LANG`. If `LANG` is likewise undefined or null, the system acts as if it were not internationalized.

If any of the locale variables has an invalid value, the system acts as if none of the variables was set.

The `LC_ALL` environment variable governs the entire locale. `LC_ALL` takes precedence over all the other environment variables that affect internationalization.

FILES

`/etc/passwd`
`/etc/shadow`
`/etc/group`
`/etc/skel`

SEE ALSO

`passwd(1)`, `users(1)`, `groupadd(1M)`, `groupdel(1M)`, `groupmod(1M)`, `logins(1M)`, `userdel(1M)`, `usermod(1M)`, `group(4)`, `passwd(4)`, `shadow(4)`.

System Administrator's Reference Manual

userdel(1M)

NAME

`userdel` - delete a user's login from the system

SYNOPSIS

`userdel` [`-r`] *login*

DESCRIPTION

The `userdel` command deletes a user's login from the system and makes the appropriate login-related changes to the system file and file system.

OPTIONS

`-r` Removes the user's home directory from the system. This directory must exist. The files and directories under the home directory will no longer be accessible following successful execution of the command.

login
A string of printable characters that specifies an existing login on the system. It may not contain a colon (`:`), or a newline (`\n`).

DIAGNOSTICS

The `userdel` command exits with one of the following values:

`0` = Success.

`2` = Invalid command syntax. A usage message for the `userdel` command is displayed.

`6` = The login to be removed does not exist.

`8` = The login to be removed is in use.

`10` = Cannot update the `/etc/group` file but the login is removed from the `/etc/passwd` file.

`12` = Cannot remove or otherwise modify the home directory.

LOCALE

The `LC_MESSAGES` environment variable governs the language in which message texts are displayed.

If `LC_MESSAGES` is undefined or is defined as the null string, it defaults to the value of `LANG`. If `LANG` is likewise undefined or null, the system acts as if it were not internationalized.

If any of the locale variables has an invalid value, the system acts as if none of the variables was set.

The `LC_ALL` environment variable governs the entire locale. `LC_ALL` takes precedence over all the other environment variables that affect internationalization.

FILES

```
/etc/passwd
/etc/shadow
/etc/group
/etc/security/ia/index
/etc/security/ia/master
/etc/security/ia/uidage
```

SEE ALSO

`passwd(1)`, `users(1)`, `groupadd(1M)`, `groupdel(1M)`, `groupmod(1M)`, `logins(1M)`, `useradd(1M)`, `usermod(1M)`, `group(4)`, `passwd(4)`, `shadow(4)`.

System Administrator's Reference Manual

usermod(1M)

NAME

usermod - modify a user's login information on the system

SYNOPSIS

usermod [–u *uid* [–o]] [–g *group*] [–G *group*[,*group* ...] [–d *dir* [–m]] [–s *shell*] [–c *comment*] [–l *new_logname*] [–f *inactive*] [–e *expire*] [–a [*operator*][*event*][,...]] *login*

DESCRIPTION

The usermod command modifies a user's login definition on the system. It changes the definition of the specified login and makes the appropriate login-related system file and file system changes.

The system file entries created with this command have a limit of 512 characters per line. Specifying long arguments to several options may exceed this limit.

OPTIONS

–u *uid*
: New UID for the user. It must be a non-negative decimal integer below MAXUID as defined in <param.h>.

–o
: This option allows the specified UID to be duplicated (non-unique).

–g *group*
: An existing group's integer ID or character-string name. It redefines the user's primary group membership.

–G *group*
: An existing group's integer ID, or character string name. It redefines the user's supplementary group membership. Duplicates between group with the –g and –G options are ignored. No more than NGROUPS_UMAX groups may be specified as defined in <param.h>.

–d *dir*
: The new home directory of the user. It defaults to *base_dir*/*login*, where *base_dir* is the base directory for the specified login home directories, and *login* is the specified login. The entry in the /etc/passwd file is modified and not the HOME directory itself. The HOME directory is only moved or modified in conjunction with the –m option.

–m
: Moves the user's home directory to the new directory specified with the –d option. If the directory already exists, it must have permissions read/write/execute for *group*, where *group* is the user's primary group.

–s *shell*
: Full pathname of the program that is used as the user's shell at login. The value of *shell* must be a valid executable file.

–c *comment*
: Any text string. It is generally a short description of the login, and is currently used as the field for the user's full name. This information is stored in the user's /etc/passwd entry.

–l *new_logname*
: A string of printable characters that specifies the new login name for the user. It may not contain a colon (:) or a newline (\n).

  Apart from the new login name, no changes are made. If the new login name is also to appear in the supplementary group, the –G option must also be used.

–e *expire*
: The date on which a login can no longer be used; after this date, no user will be able to access this login. (This option is useful for creating temporary logins.) You may type the value of the argument

expire (which is a date) in various formats (included in `/etc/datemsk`). For example, you may enter `10/6/01` or `October 6, 2001`. A value of `""` defeats the status of the expired date.

Note:

The date is indicated in Greenwich Mean Time (GMT) and may therefore differ by as much as a day in other time zones. For example, if an expiry date only comes into effect in Central European Time CET at 1 am and not at midnight, which would correspond to GMT.

`–f` *inactive*

The maximum number of days allowed between uses of a login ID before that login ID is declared invalid. Normal values are positive integers. A value of `0` defeats the status.

`–a` [*operator*][*event*][,...]

Set the user's audit mask based on the event(s) specified. The following operators are available: `+` to add the events to the user's audit mask, `–` to delete the events to the user's audit mask, `=` to replace the user's audit mask. If no operator is specified, the `=` operator (replace) is assumed. This option is only valid if the auditing subsystem is installed.

login

A string of printable characters that specifies the existing login name of the user. It must exist and may not contain a colon (`:`), or a newline (`\n`).

DIAGNOSTICS

The `usermod` command exits with one of the following values:

`0` = The command was executed successfully.

`2` = The command syntax was invalid. A usage message for the `usermod` command is displayed.

`3` = An invalid argument was provided to an option.

`4` = The *uid* given with the `–u` option is already in use.

`6` = The login to be modified does not exist or *group* does not exist.

`8` = The login to be modified is in use.

`9` = The *new_logname* is already in use.

`10` = Cannot update the `/etc/group` file. Other update requests will be implemented.

`11` = Insufficient space to move the home directory (`–m` option). Other update requests will be implemented.

`12` = Unable to complete the move of the home directory to the new home directory.

LOCALE

The `LC_MESSAGES` environment variable governs the language in which message texts are displayed.

If `LC_MESSAGES` is undefined or is defined as the null string, it defaults to the value of `LANG`. If `LANG` is likewise undefined or null, the system acts as if it were not internationalized.

If any of the locale variables has an invalid value, the system acts as if none of the variables was set.

The `LC_ALL` environment variable governs the entire locale. `LC_ALL` takes precedence over all the other environment variables that affect internationalization.

FILES

`/etc/passwd`

`/etc/shadow`

`/etc/group`

SEE ALSO

`passwd(1)`, `users(1)`, `groupadd(1M)`, `groupdel(1M)`, `groupmod(1M)`, `logins(1M)`, `useradd(1M)`, `userdel(1M)`, `group(4)`, `passwd(4)`, `shadow(4)`.

System Administrator's Reference Manual

usvadm(1M)
RM600 only

NAME

`usvadm -` configuration of uninterruptible power supply (UPS)

SYNOPSIS

`usvadm –p` *board_boardno,portno* [`–m` *type*] [`–c`] [`–h` *hold_time*] [`–s` *shutdown_time*]

DESCRIPTION

The uninterruptible power supply (UPS) is configured using `usvadm`, by entering the configuration parameters in the `/etc/rc2.d/S98usv` file.

Once the configuration parameters have been entered, you must then activate the UPS daemon by rebooting the system, or by executing the `/etc/rc2.d/S98usv` file. In both instances, if there is no UPS connection, the system is immediately shut down in system administrator mode (init state 1).

OPTIONS

`–p` *board_boardno,portno*
: The board name, board number, and the number of an available TTY port (e.g. `itp0,15`). Disable any listeners that are connected to the TTY interface. This can be done using the sysadm interface (monitoring programs for ports, administration of port facilities, removal of port facilities).

`–m` *type*
: The type of UPS installed. *type* is a three-digit decimal number (e.g. `105` or `310`).

`–c`
: By selecting this option, the UPS daemon `usvd` will be started at the next system startup. This option must be selected to enable the daemon to monitor UPS, and can only be omitted in exceptional cases, where the system is to be operated without being monitored by the UPS daemon (e.g. for UPS maintenance). If you select this option, and there is no UPS connection, the system will be immediately shut down in system administrator mode at the next system startup (init state 1).

`–h` *hold_time*
: This option allows you to specify the battery reserve time in seconds (default: 300 seconds). The battery reserve time is the period of time between the loss of power and the start of the `/etc/powerfail` script. Note that the length of time for which power can be maintained depends on the power consumption of the connected devices, the UPS type selected and the battery status.

`–s` *shutdown_time*
: This option allows you to specify the UPS shutdown time in seconds (default: 50 seconds). The shutdown time is the period of time between the start of the `/etc/powerfail` shell script and the UPS shutdown. Note that this time period should be long enough to ensure that all applications are terminated correctly and that the system is shut down securely.

EXAMPLES

```
usvadm -p itp0,15 -m 105 -c -h 400 -s 120
usvadm -p chase1,6 -m 105 -c -h 400 -s 120
usvadm -p motherboard0,5 -m 105 -c -h 400 -s 120
usvadm -p csi0,2 -m 105 -c -h 400 -s 120
usvadm -p tca2,5 -m 105 -c -h 400 -s 120
```

FILES

```
/etc/powerfail
/etc/rc2.d/S98usv
```

SEE ALSO

usvd(1M) .

System Administrator's Reference Manual

usvd(1M)
RM600 only

NAME

`usvd -` daemon for uninterruptible power supply (UPS)

SYNOPSIS

`/etc/rc2.d/S98usv [ start | stop | status ]`

DESCRIPTION

`usvd` is the UPS monitoring daemon. It is usually called at boot time from the `/etc/rc2.d/S98usv` file. It establishes a communication connection to `/dev/term/usv`, where `/dev/term/usv` is a link to any TTY interface. This TTY is physically connected to the communications port on the UPS via an RS232 cable.

The daemon polls UPS every 60 seconds. If the UPS reports a power failure, the `usvd` daemon then checks the UPS battery voltage every 30 seconds. If the voltage falls below a given limit, or the battery reserve time expires, the UPS daemon calls the `/etc/powerfail` shell script and then shuts down the system (`/etc/shutdown –g –g0 i0`). UPS is then shut down as well.

System administrators can make specific entries in the `/etc/powerfail` shell script for reacting to a power failure. The time interval between calling the `/etc/powerfail` shell script and the UPS shutdown is determined by the shutdown time specified during configuration in the `/etc/rc2.d/S98usv` file (default: 50 seconds).

For information on how to switch on the UPS after a power failure, please consult the literature provided. In general, you should always switch on the UPS after a power failure before rebooting your system.

If there is a problem during system startup, or communication with UPS is lost, the UPS daemon shuts down the system after 60 seconds [init state 1 (`shutdown –g –y0 –i1`)]. The system administrator can now check the hardware and reboot the system (`init 2`).

Before `usvd` can establish a connection to `/dev/term/usv`, the following steps must be taken:

- Choose the TTY to which the UPS is to be connected. Using the RS232 cable supplied with the UPS, connect the TTY to the UPS communication port.
- Use `pkginfo SIusv` to check whether the UPS package has been installed, and if not, install it with `pkgadd SIusv`.
- Configure the UPS using the command `usvadm`.
- Activate the UPS daemon by rebooting the system, or by executing the `/etc/rc2.d/S98usv` file. If there is no UPS connection, the system is immediately shut down in system administrator mode, in both cases (`init state 1`).

Check the system configuration by carrying out the following steps: First, write the current UPS daemon status to the log file. Then, evaluate the log file using the command `log3display –s USV`.

Entries are made in the log file for the following events:

USV: powerfail (*)
USV: error (*)
USV: powerback (*)
USV: `usvd` shutdown
USV: daemon error
USV: hardware status
USV: communication error
USV: cannot reach tty line
USV: illegal message from usv
USV: daemon abnormally died
USV: unknown event

USV: in bypass state
USV: in battery operation because of power breakdown
(*) see status information in log file

OPTIONS

`start`
activates the UPS daemon

`stop` deactivates the UPS daemon

`status`
writes the current status of the UPS daemon to the log file.

DIAGNOSTICS

Messages are output on the system console and to the log file.

```
USV daemon: usvd started
USV daemon: POWERFAIL arrived battery_hold_time
USV daemon: POWERFAIL since xx sec. vbatt xx V battery_rest_time xx sec.
USV daemon: POWERBACK arrived after xx sec.
USV daemon: Bit Vbattlowwarn set
USV daemon: Internal error in USV  SYSTEM going to halt
USV daemon: Warning: USV phase is broken
USV daemon: Warning: USV in short BATTERY mode
USV daemon: Warning: USV in BYPASS mode
            You are not protected against a power supply breakdown!
USV daemon: Illegal message from USV  SYSTEM going to halt
USV daemon: /etc/powerfail called
            SYSTEM GOING TO HALT IMMEDIATELY
USV daemon: USV power switch off in xx sec.
USV daemon: >>> EMERGENCY HALT - reboot(RB_HALT) <<<
USV daemon: Unable to communicate with the USV
            Please check your system configuration
            GOING TO INIT STATE 1
USV daemon: Communication with USV interrupted
            Please check your system configuration
USV daemon: Cannot stop the system
USV daemon: Abnormal end (aboard by a signal)
USV daemon: Time system call failed
USV daemon: Logging failed
```

FILES

```
/etc/powerfail
/etc/rc2.d/S98usv
/etc/rc1.d/K60usv
```

SEE ALSO

`usvadm`(1M) .

System Administrator's Reference Manual

viewdict(1M)

NAME

`viewdict` - view DBM dictionary

SYNOPSIS

`viewdict` *dbm_dictionary*

DESCRIPTION

`viewdict` reads a DBM database called *dbm_dictionary* and outputs its contents in readable format to standard output.

FILES

dbm_dictionary`.dir`
: DBM data base containing a bit map

dbm_dictionary`.pag`
: DBM data base containing all data

SEE ALSO

`passwd`(1), `makedict`(1M), `dbm`(3C).

System Administrator's Reference Manual

vigrp(1M)

NAME

vigrp - edit the group file

SYNOPSIS

vigrp [–dns] [–m *n*] [–w *n*] [*name*]

DESCRIPTION

vigrp edits the group file *name* (/etc/group by default), folding long lines at a group member boundary. A "\" at the end of a line indicates that a group member list continues on the next line.

vigrp is particularly useful for viewing and modifying group files having lines longer than some text editors are able to handle. The vi editor will be used unless the environment variable EDITOR indicates an alternative editor. If the group file is currently being edited or the user does not have write permission for the relevant directory, the file will be opened read-only. After leaving the editor, if the file was modified and the –n option was not given, the file will be checked with grpck(1M) .

OPTIONS

–d Displays duplicate entries.

–m *n* Displays a maximum of *n* group members per line.

–n Performs no check on the modified file with grpck.

–s Sorts the entries by group number and group member, removing duplicates.

–w *n* Displays a maximum line width of *n* characters (default 80).

FILES

The contents of *name* are copied to /tmp/group.XXXXXX, which is then passed to the editor with long lines folded. If changes were made, the folded lines are concatenated and written to *name*.tmp, grpck is run over the file (if –n is not specified), *name* is renamed to o*name*, and *name*.tmp is renamed to *name*.

SEE ALSO

grpck(1M) , group(4) .

System Administrator's Reference Manual

vmstat(1M)

NAME

`vmstat -` report virtual memory statistics

SYNOPSIS

`/usr/ucb/vmstat` [`–d`|`–S`] [*interval* [*count*]]

`/usr/ucb/vmstat –f –m –s`

DESCRIPTION

`vmstat` examines the system and normally reports certain statistics kept about process, virtual memory, disk, trap and cpu activity.

OPTIONS

No option specified:

`vmstat` reports (in the fist line) a summary of the virtual memory activity since the system has booted.

`–d` Reports the disk assignments (e.g. `d0 = ios0/sdisk000`, `d1 = ios0/sdisk010` etc.).

`–S` Prints the "si" and "so" format fields, instead of the "re" and "at" format fields (see below).

`–f` Reports on the number of `forks` since system startup.

`–m` Reports on the number of physical pages currently being used by the operating system.

`–s` Prints the contents of the `sum` structure, giving the total number of several kinds of paging-related events which have occurred since boot.

interval

If *interval* is specified, then successive lines (after the first) are summaries over the last *interval* seconds.

`vmstat 5` will print what the system is doing every five seconds; a print interval of 5 seconds is recommended as statistics are tested in the system every 5 seconds. Other statistics vary every second; running the output for a while will make it apparent which statistics are recomputed every second.

count

If a *count* is given, the statistics are repeated *count* times.

Format fields

`procs`

Information about numbers of processes in various states.

`r` In run queue.

`b` Blocked for resources (I/O, paging, etc.).

`w` Operable process or short sleeper (< 20 secs) but swapped.

`memory`

Information about the usage of virtual and real memory. Virtual pages are considered active if they belong to processes which are running or have run in the last 20 seconds. A page here is 1024 bytes.

`avm` Active virtual pages.

`fre` Size of the free list.

`page` Information about page faults and paging activity. These are averaged each five seconds, and given in units per second.

`re` Page reclaims.

si Swap ins (when called with the –S option).

at Pages attached (found in free list).

so Swap outs (when called with the –S option).

pi Pages paged in.

po Pages paged out.

fr Pages freed per second.

sr Pages scanned by clock algorithm, per second.

cpu(s)

Breakdown of percentage usage of CPU time as well as the number of interrupts, system calls and context switches per second.

in Device interrupts fielded by this cpu per second.

syc System calls per second.

cs Number of context switches per second.

us Percentage of cpu time spent in user mode.

sy Percentage of cpu time spent in system mode (other than idle).

id Percentage of cpu time spent in idle or I/O wait mode.

io Percentage of cpu time spent in I/O wait mode.

The following line(s) contains/contain I/O statistics for individual controllers as well as global I/O statistics.

EXAMPLES

Calls (without output):

```
vmstat
vmstat –d
vmstat –S 5 3
vmstat 5 10
vmstat –fs
```

Calls (with output):

```
vmstat
 procs      memory              page                    cpu(s)
 r b w    avm    fre    re  at  pi po fr sr     in  syc   cs us sy id io
 2 0 0   8484 304860     0  32   0  1  1  0     18  161   22  1 11 81  7
0s00: 2     0s03: 2     0s04: 1     0s05: 0     0s06: 2     Total: 6
```

SEE ALSO

scstat(8).

System Administrator's Reference Manual

volcopy(1M)

NAME

`volcopy(generic)` - make literal copy of file system

SYNOPSIS

`volcopy` [`–F` *FSType*] [`–V`] [`–a`|`–sy`] [`–buf`] [`–bpi` *density*] [`–feet` *size*] [`–reel` *num*] [`–o` *specific_options*] *operands*

DESCRIPTION

`volcopy` makes a literal copy of the file system.

There are two types of option: *generic_options* and *specific_options*. *generic_options* indicate options that are supported not only by the generic but also by the *FST*-specific module of the command (if that command exists). *specific_options* indicate suboptions that are supported only by the *FST*-specific module of the command. They are specified in a comma-separated list and/or as keyword-attribute pairs for interpretation by the *FST*-specific module of the command. These suboptions are specified via the `–o` option. The options are of two types: *generic_options* and *specific_options*. *generic_options* indicate options that are supported by the *FSType*-specific module of the command (if that command exists), and *specific_options* indicate suboptions specified in a comma-separated list of suboptions and/or keyword-attribute pairs for interpretation by the *FSType*-specific module of the command. The latter are specified via the `–o` option.

operands generally include the device and volume names and are file system specific. A detailed description of the *operands* can be found in the *FSType*-specific man pages for `volcopy`.

OPTIONS

The *generic options* are:

`–F` Specifies the *FSType* on which to operate. The *FSType* should either be specified here or be determinable from `\/etc\/vfstab` by matching the *operands* with an entry in the table.

`–V` Echoes the complete command line, but does not execute the command. The command line is generated by using the options and arguments provided by the user and adding to them information derived from `/etc/vfstab`. This option should be used to verify and validate the command line. When used with the *FSType* option, it passes the `–V` to the *FSType*-specific module for interpretation.

`–a` Invokes a verification sequence ("yes") requiring a positive operator response instead of the standard 10 second delay before the copy is made.

`–s` Inverse of `–a`. A 10 second delay is given after the from/to devices are printed (followed by a `?`). During this time the command can be aborted by entering a delete (DEL).

`–y` Assumes a "yes" response to all questions.

The following *generic options* are used only with reel tapes: (not applicable to cartridge tape):

`–bpi` *density*
Bits-per-inch (i.e. `800/1600/6250`).

`–feet` *size*
Size of reel in feet (i.e. `1200/2400`).

`–reel` *num*
Beginning reel number for a restarted copy.

`–buf` Uses double buffered I/O.

`–o` Specifies *FSType*-specific options.

NOTE

This command may not be supported for all FSTypes.

FILES

`/etc/vfstab`
: list of default parameters for each file system

SEE ALSO

`vfstab`(4) .

Manual pages for the FSType-specific modules of `volcopy`.

System Administrator's Reference Manual

volcopy(1M-ufs)

NAME

`volcopy(ufs)` - make a literal copy of a ufs file system

SYNOPSIS

`volcopy [ -F ufs ] [` *generic_options* `]` *fsname srcdevice volname1 destdevice volname2*

DESCRIPTION

`volcopy` makes a literal copy of the `ufs` file system using a blocksize matched to the device (raw device only).

OPTIONS

`-F ufs`
: Specifies the `ufs`-FSType.

generic_options
: Options supported by the generic `volcopy` command.

The *fsname* argument is the same as the *fsname* specified with `labelit`(1M-ufs) . If there is no *fsname*, then `""` (empty character string) must be specified.

The *srcdevice* or *destdevice* should be the physical disk section or tape (e.g. `/dev/ios0/rsdisk000s3`, `/dev/ios0/rstape00`, etc.).

The *volname* is the physical volume name. Such label names are limited to six or fewer characters. *volname* may be "-" to use the existing volume name.

srcdevice and *volname1* are the device and volume from which the copy of the file system is being extracted. *destdevice* and *volname2* are the target device and volume.

fsname and *volname* are recorded in the superblock (`char fsname[6], volname[6];`).

NOTE

`volcopy` does not support tape-to-tape copying. Use `dd`(1) for tape-to-tape copying.

FILES

`/var/adm/filesave.log`
: arecord of file systems/volumes copied

SEE ALSO

`cpio`(1), `dd`(1), `labelit`(1M), generic `volcopy`(1M), `fs`(4-ufs) .

volcopy(1M-vxfs)
VxFS

NAME

`volcopy(vxfs)` - make a literal copy of a Veritas File System (VxFS)

SYNOPSIS

`volcopy` [`–F vxfs`] [*generic_options*] [`–e`] *fsname srcdevice volname1 destdevice volname2*

DESCRIPTION

This manual page is an addendum to the `volcopy(1M)` manual page. It describes the Veritas File System (VxFS) specific features of the `volcopy` command.

`volcopy` makes a literal copy of the Veritas file system using a block size matched to the device.

OPTIONS

`–F vxfs`
Specifies the Veritas file system type (*FSType*).

generic_options
Options supported by the generic `volcopy` command.

`–e` Processes until the end of tape, then prompts for the next tape in the sequence.

If the `–e` option is not selected, the program requests length and density information, if this information is not already provided on the command line, or if this information is not recorded on an input tape label. If the file system is too large to fit on one reel, `volcopy` prompts for additional reels. Labels of all reels are checked.

Tapes may be mounted alternately on two or more drives. If `volcopy` is interrupted, the user is prompted either to quit or escape to the command interpreter. If the user chooses to escape to the command interpreter, other operations can be performed [such as `labelit(1M)`]. The user can then return to `volcopy` by exiting the command interpreter. Note that the `–e` and `–feet` options are mutually exclusive.

fsname
The mounted name of the file system being copied (for example, `/` or `/usr`).

srcdevice
The device from which the copy of the file system is being extracted.

volname1
The volume from which the copy of the file system is being extracted.

destdevice
The target device to which the file system will be written.

volname2
The target volume to which the file system will be written.

srcdevice or *destdevice* are specified as the disk partition or tape (for example, `/dev/ios0/rsdisk00`, `/dev/rmt/???`).

volname1 or *volname2* are the physical volume names. Such label names contain up to six characters. *volname** may be "–" to use the existing volume name. *fsname* and *volname* are recorded in the super block.

NOTES

`volcopy` does not support tape-to-tape copying. Use `dd(1)` for tape-to-tape copying.

FILES

`/var/adm/filesave.log`
arecord of file systems/volumes copied

SEE ALSO

`cpio`(1), `dd`(1), `labelit`(1M), `volcopy`(1M), `fs`(4-vxfs).

System Administrator's Reference Manual

vtgetty(1M)
RM400 only

NAME

`vtgetty` - set terminal type, modes, speed, and line discipline

SYNOPSIS

`/etc/vtgetty` [`–h`] [`–t` *timeout*] *line* [*speed* [*type* [*linedisc*]]]

DESCRIPTION

The `vtgetty` command is a program invoked by `init`(1M) . It is the second process in the series (`init-vtgetty-getty-login-shell`). The `/etc/getty` process will ultimately connect a user with the Reliant UNIX system. `vtgetty` can be executed only by the superuser (a process with the user-ID of root).

The command options are identical to those of `getty`(1M) .

Initially, `vtgetty` opens the device and determines if any virtual terminals (vts) are open for that device. If there are active vts, the user will be prompted to determine if the vts should be closed automatically or manually when the user logs out.

If the automatic option is selected, `vtgetty` will send the signals, `SIGHUP` and `SIGTERM`, to each open vt.

It will then wait 3 seconds and send a `SIGKILL` signal to the vts to ensure that all the vts are terminated.

If the manual closure option is selected, the highest numbered vt will be activated and the user can manually close the vt. This will be repeated until all open vts are manually closed.

DIAGNOSTICS

`vtgetty` will fail under the following conditions:

- There is no memory available.
- It cannot open the device it was given.
- It cannot convert from a file descriptor to a file pointer.
- It cannot get the file status [`stat`(2)] of the device it was given.
- An `ioctl`(2) call fails.

FILES

`/etc/gettydefs`

SEE ALSO

`kill`(1) , `tty`(1) , `getty`(1M) , `init`(1M) , `ioctl`(2) , `stat`(2) , `gettydefs`(4) , `inittab`(4) .

System Administrator's Reference Manual

vtlmgr(1M)
RM400 only

NAME

`vtlmgr` - monitor and open virtual terminals

SYNOPSIS

`vtlmgr` [`–k`]

DESCRIPTION

When you invoke the `vtlmgr` command (usually from within your `.profile`), it places itself in the background and monitors `/dev/vtmon` for signals from the keyboard/display driver to open new virtual terminals.

The `–k` option sends a `SIGHUP` signal to all open virtual terminals when you log off (by entering <CTRL-D> from your home virtual terminal). This automatically closes, if possible, existing virtual terminals. For virtual terminals that cannot be closed automatically, you are asked if you want to close them manually.

After running `vtlmgr`, you open new virtual terminals and then switch between them by entering a hot-key sequence, specifically:

`ALT - SYS-REQ` *key*

where *key* is either a function key whose number corresponds to the number of the virtual terminal to switch to, for example, pressing F1 switches you to `/dev/vt01` (virtual terminal 01), pressing F2 switches you to `/dev/vt02` (virtual terminal 02), and so forth, or one of the letters in the following table:

key	Interpretation
`h`	home virtual terminal (`/dev/vt00`)
`n`	next virtual terminal
`p`	previous virtual terminal
`f`	force a switch to a virtual terminal

Use the `f` key only when the current virtual terminal is essentially locked up or stuck in graphics mode. This will cause the virtual terminal to be reset to a sane text state and all processes associated with the virtual terminal will be killed.

When the hot-key sequence is entered, the executable program pointed to by the `$SHELL` variable is executed in the new virtual terminal. If `$SHELL` is NULL or pointing to a program which is not executable, `/bin/sh` is executed. The newly opened virtual terminal inherits the environment in effect when the `vtlmgr` command is invoked.

You may perform setup on each new virtual terminal as it is created by `vtlmgr` through the `.vtlrc` file. This file should be in your home directory. Its contents are a shell script that will be run by `/bin/sh` before the shell prompt is displayed. In this way it is similar to your `.profile` file. However, you may not set and export environment variables to the shell for the virtual terminal because a different shell runs the `.vtlrc` shell script.

The system administrator can control how many virtual terminals are available by setting a parameter in the file `/etc/default/workstations`. Virtual terminals 0–8 are configured by default and the default keyboard map makes up to 13 virtual terminals available (i.e. an additional 4 virtual terminals can readily be defined within the default settings). The default virtual terminals are the home terminal and one corresponding to each function key. An application can make two more available to the end-user (by reprogramming the keyboard map), or can reserve the last two for programmatic use only, making 15 virtual terminals available in all.

Note that processes that are no longer visible may still be continuing. Standard output is directed to the current virtual terminal's screen. For example, you can issue a `cat` command on one virtual terminal, switch

to another virtual terminal to start an application, and then switch to another to do an edit. The `cat` output will be lost if the virtual terminal scrolls the data off the screen unless you initially redirect the output to a file.

DIAGNOSTICS

The `vtlmgr` command will fail under the following conditions:

- An illegal option is specified.
- The device cannot be opened.
- The command is invoked from a remote terminal.
- `/dev/vtmon` cannot be opened.
- `$SHELL` is set and is not executable.
- `$SHELL` is not set and `/bin/sh` cannot be invoked.

SEE ALSO

`vtgetty`(1M) , `keyboard`(7) .

System Administrator's Reference Manual

vxdump(1M)
VxFS

NAME

`vxdump` - incremental file system dump

SYNOPSIS

`/usr/lib/fs/vxfs/vxdump` [*options*] *filesystem*

DESCRIPTION

`vxdump` back up all files in *filesystem*, or files changed after a certain date, to an output device or file; *options* is a string that specifies `vxdump` options, as shown below; *filesystem* is a character special device (for example, `/dev/ios0/rsdisk000s4`) or a mount point for a file system (for example, `/usr`).

If no *options* are given, the default is `9–u` and the default output device is assumed to be a 2400 foot, 6250 bpi 9-track tape.

If the `–c` option is specified, the output device is assumed to be a QIC-320 cartridge tape.

If the `–D` option is specified, the output device is assumed to be a DAT (Digital Audio Tape) device.

Options `–c`, `–d`, `–D`, and `–s` are only used to calculate tape size. The `–M` option is used to specify megabytes per tape, and may be used with any type of tape. The `–M` option is much simpler than the above options, provided the tape capacity is known.

`vxdump` does not handle EOT (end of tape) gracefully. Unlike other backup utilities, such as `cpio`, that detect the EOT and then prompt the user to insert another tape before continuing the backup, `vxdump` will issue a write error upon reaching the EOT and will attempt to restart the backup. It is, therefore, important to supply `vxdump` with accurate information about the tape capacity with the `–M` option, or about tape type, density, and length. This will allow `vxdump` to determine how much data to write to each tape before asking for the next tape. (See **EXAMPLES** below.) Since `vxdump` doesn't handle EOT, it is important to be aware that if you over-estimate how much data fits on the tape, the last of the multiple dumps on the tape will be incomplete. You can store multiple dumps on one tape by writing to the no-rewind device, although you may need to restart the last dump when it hits EOT.

Example:

```
vxdump 0uf /dev/ios0/rstape000hn /usr
vxdump 0uf /dev/ios0/rstape000h /
```

The above commands write a level `0` dump of `/usr` and a level `0` dump of `/` (root) to the same tape. Notice the use of the no-rewind device in the dump of `/usr`. Multiple dumps can be recovered using the `–s` option in `vxrestore`(1M) .

Use the following guidelines, keeping in mind that file systems with many small files (<2 KB) will use more tape, and that use of the `–M` option is recommended in lieu of other options which determine tape size (`–c`, `–d`, `–D`, and `–s`):

Tape	Mode	Type	Length	Capacity
9-track	–	2400	6250ft	~140 MB
9-track	–	2400	1600ft	~25 MB
Streamer:				
QIC-320	–	600	–	~120 MB
QIC-120	–	120M	620ft	~105 MB
QIC-150	–	150M	620ft	~135 MB
QIC-525	–	525M	620ft	~475 MB

DAT/DDS-1	–	–	60m	~1200 MB
DAT/DDS-1	–	–	90m	~1800 MB
DAT/DDS-2	–	–	120m	~3600 MB
8mm Tape	EXB-8200	–	15m	~265 MB
(Exabyte)	EXB-8500	–	15m	~530 MB

Notes:

- EXB-8200 and EXB-8500 are different write densities for working with 8 mm tapes (low- and high-density).
- If a "device with compression" is being used (e.g. `/dev/ios0/rstape004c`), this increases the capacity by a factor of 1.6 – 2.0 (maximum).
- The capacity of all "tapes" is approx. 10% below the specified sizes (e.g. 150 MB streamer can record approx. 135 MB data). The reduction in the specified value takes into account any tape errors and automatic tape leader for lower transfer rates.
- Specific details on all tape types (e.g. how a 150 MB streamer behaves in a 525 MB streamer drive) can be found in the **TAPE NOTES** for `stape(7)`.

OPTIONS

`0–9` Dump level. All files in the *filesystem* that have been modified since the last `vxdump` at a lower dump level are copied to the volume. For instance, if you did a level `2` dump on Monday, followed by a level `4` dump on Tuesday, a subsequent level `3` dump on Wednesday would contain all files modified or added since the level `2` (Monday) backup. A level `0` dump copies the entire *filesystem* to the dump volume.

`–b` *factor*
Blocking factor. Specifies the blocking factor for tape writes. The default is `64` blocks per write. A tape block is 512 bytes. Note that the blocking factor must be a multiple of 4.

`–c` Cartridge. Uses a QIC-320 instead of the standard half-inch reel. The length is set to 560 feet. This option is incompatible with the `–d` and `–s` options. If the tape capacity is known, the `–M` option is recommended instead of the `–c` option.

`–d` *bpi*
Tape density. Meaningful only for 9-track tapes. The density of the tape, expressed in bits per inch, is taken from *bpi*. This is used to keep a running tab on the amount of tape used per reel. The default density is `6250`.

`–D` *mins*
DAT. Uses DAT (Digital Audio Tape) instead of the standard half-inch reel. *mins* is the number of minutes on the DAT tape. This option is incompatible with the `–d` and `–s` options.

`–f` *dump-file*
Dump file. Uses *dump-file* as the output device or file to dump to, instead of the default (`/dev/rmt8`). If *dump-file* is specified as `–`, dump to the standard output. Note that *dump-file* can be the name of a disk file (for example, `/tmp/dump.output`). This creates a dump image on disk. This can then be restored (for example, `vxrestore xuf /tmp/dump.output`), and transferred to another system or written to tape using another utility, such as `cpio`.

`–M` *megabytes*
Maximum amount of data to write to each tape, given in megabytes (10^6). This option determines the maximum amount of data to write to each tape before `vxdump` will ask for another tape. This can be used with QIC, DAT, 9-track, or any type of tape device. The maximum *megabytes* specified should be somewhat less than the capacity of the tape, to allow for retries on bad sections of tape. This option overrides the `–c`, `–d`, `–D`, and `–s` options.

–n Notifies all operators in the operator group that vxdump requires attention by sending messages to their terminals, in a manner similar to that used by the wall command.

–s *size*
Specifies the *size* (in feet) of the volume being dumped to. When the specified *size* is reached, vxdump waits for you to change the volume. The default *size* for 9-track tape is 2300 (feet), which is slightly less than the actual physical size of the tape. This allows for bad sectors on the tape and guards against tape overrun.

–t *tracks*
Specifies the number of tracks for a cartridge tape. The default is 9 tracks. The –t option is not compatible with the –D option.

–u Updates the dump record. For each file system successfully dumped, adds an entry to the file /etc/dumpdates, that includes the file system name, date, and dump level. This file can be edited by the superuser.

–W The –W option lists all file systems that appear in /etc/dumpdates and /etc/fstab, along with information about their most recent dump dates and levels. If the –W option is set, all other options are ignored, and vxdump exits immediately.

EXAMPLES

The following examples describe possible command lines using vxdump options:

vxdump 0u /usr
Perform a level 0 dump of /usr to /dev/rmt8, which is a 6250 bpi 9-track tape drive with a 2400 foot tape mounted on it; update /etc/dumpdates.

vxdump 0Muf 320 /dev/rmt8 /usr
Perform a level 0 dump of /usr to /dev/rmt8, which writes a maximum of 320 MB to each tape before asking for the next tape. The type of tape device does not matter, but you must know the tape capacity.

vxdump 0udf 1600 /dev/rmt0 /usr
Perform a level 0 dump of /usr to /dev/rmt0, which is a 1600 bpi 9-track tape drive with a 2400 foot tape mounted on it; update /etc/dumpdates.

vxdump 0usf 1200 /dev/ios0/rstape001h /usr
Perform a level 0 dump of /usr to /dev/ios0/rstape001h, which is a 6250 bpi 9-track tape drive with a 1200 foot tape mounted on it; update /etc/dumpdates.

vxdump 2ucf /dev/ios0/rstape000 /home
Perform a level 2 dump of /home to /dev/ios0/rstape000, which is a QIC-320 cartridge tape; update /etc/dumpdates.

vxdump 0Df 120 /dev/ios0/rstape000 /usr
Perform a level 0 dump of /usr to /dev/ios0/rstape000, which is a DAT 60-meter (120 minute) tape; do not update /etc/dumpdates.

NOTES

Fewer than 32 read errors on the file system are ignored.

Each tape requires a new process, so parent processes for tapes already written just hang around until the entire archive is written.

It is recommended that incremental dumps also be performed with the system running in singleuser mode.

For a multiple-tape dump. always use tapes of exactly the same capacity, to avoid hitting EOT on a shorter tape.

For simplicity, if the tape size in megabytes is known, the –M option is recommended, instead of the –c, –d, –D, or –s options.

If you do multiple dumps to a "no-rewind" tape device file, the maximum capacity to write to each tape will not protect vxdump from eventually hitting EOT. This may require that you restart the last (failed) dump on a new

tape.

SECURITY NOTES

Only a superuser can do a dump of a file system. This prevents users from being able to gain access to files with selective read permissions.

There are special considerations for specifying the remote username in the *dump-file* to the `-f` option. In order to specify a user other than the one who invoked the remote dump or restore utility, the user who invoked the utility must have permission from the remote user. This is accomplished by making the appropriate entry in the `.rhosts` file which resides in the remote user's home directory. The entry should include the originating hostname and username (which will always be "root" since only a superuser can dump a file system).

In addition to the normal security checks, the security parameters can affect the use of the remote dump and restore utilities.

If the `AUTO_LOGIN_MIN_UID` parameter is set to a positive number on the remote host system, then users with user ids less than that number cannot use automatic remote access.

If the `DISABLE_RHOSTS` parameter is set to "yes" on the remote host system, then any `.rhosts` files will be ignored on the remote host system. If the `DISABLE_RHOSTS` parameter is set to "no" on the remote host system, then any `.rhosts` files must be writable only by the owner.

NOTES

The `-s` option should not be used with the `-c` option. If these options are used together, the number of feet specified will be multiplied by 4 resulting in erroneous capacities per tape.

`vxdump` does not gracefully handle EOT.

FILES

`/dev/rmt8`
: default device to dump to

`/etc/dumpdates`
: dump date record

`/etc/group`
: to find group operator

SEE ALSO

`tar`(1), `shutdown`(1M), `vxrestore`(1M), `wall`(1M).

System Administrator's Reference Manual

vxrestore(1M)
VxFS

NAME

`vxrestore -` incremental file system restore

SYNOPSIS

`vxrestore` [`-irRtxdhmvy`] [`-b` *factor*] [`-f` *dump-file*] [`-s` *n*] [*filename ...*]

DESCRIPTION

`vxrestore` restores files from backup tapes created with the `vxdump` command using at least one of the options listed below, along with any modifiers and arguments you supply. Remaining arguments to `vxrestore` are the names of files (or directories) whose files are to be restored to disk. Unless the `-h` modifier is in effect, a directory name refers to the files it contains, and (recursively) its subdirectories and the files they contain.

OPTIONS

`-i` Interactive. After reading in the directory information from the tape, `vxrestore` invokes an interactive interface that allows you to browse through the dump tape's directory hierarchy and select individual files to be extracted. See **Interactive Commands**, below, for a description of available commands.

`-r` Restores the entire tape and loads the tape's full contents into the current directory. This option should be used only to restore a complete dump tape onto a clear filesystem, or to restore an incremental dump tape after a full "level `0`" restore.

`-R` Resumes restoring. `vxrestore` requests a particular tape of a multivolume set from which to resume a full restore (see the `-r` option above). This allows `vxrestore` to start from a checkpoint when it is interrupted in the middle of a full restore.

`-t` Table of contents. Lists each *filename* that appears on the tape. If no *filename* argument is given, the root directory is listed. This results in a list of all files on the tape, unless the `-h` modifier is in effect.

`-x` Extracts the named files from the tape. If a named file matches a directory whose contents were written onto the tape, and the `-h` modifier is not in effect, the directory is recursively extracted. The owner, modification time, and mode are restored (if possible). If no *filename* argument is given, the root directory is extracted. This results in the entire tape being extracted unless the `-h` modifier is in effect.

`-d` Debug. Turns on debugging output.

`-h` Extracts the actual directory, rather than the files that it references. This prevents hierarchical restoration of complete subtrees from the tape.

`-m` Extracts by inode numbers rather than by file name to avoid regenerating complete path names. This is useful if only a few files are being extracted.

`-v` Verbose. `vxrestore` displays the name of each file it restores, preceded by its file type.

`-y` Does not ask whether to abort the restore in the event of tape errors. `vxrestore` tries to skip over the bad tape block(s) and continue as best it can.

`-b` *factor*
Blocking factor. Specifies the blocking factor for tape reads.

Note: A tape block is 512 bytes.

`-e` *extent_op*
Specifies how to handle extent attribute information. This option specifies the required persistence of extent attributes when restoring files which had preallocated space or fixed extent sizes. Valid values for *extent_op* are:

`warn` Issues a warning message if extent attribute information cannot be kept (the default behavior if

this option is not specified).

`force`
Fails the file restore if extent attribute information cannot be kept.

`ignore`
Ignores extent attribute information.

`-f` *dump-file*
Uses *dump-file* instead of `/dev/rmt?` as the file to restore from. If *dump-file* is specified as "-", `vxrestore` reads from the standard input. This allows `vxdump(1M)` and `vxrestore` to be used in a pipeline to dump and restore a file system:

```
vxdump 0f - /dev/ios0/rsdisk000s3 | (cd /mnt; vxrestore xf -)
```

`-s` *n* Skips to the *n*'th file when there are multiple dump files on the same tape. For example, the command:

```
vxrestore xfs /dev/ios0/rstape000h 5
```

would position you at the fifth file on the tape.

Interactive Commands

`vxrestore` enters interactive mode when invoked with the `-i` option. Interactive commands are reminiscent of the shell. For those commands that accept an argument, the default is the current directory.

`ls` [*directory*]
Lists files in *directory* or the current directory, represented by a "`.`" (period). Directories are appended with a "`/`" (backslash). Entries marked for extraction are prefixed with a "`*`" (asterisk). If the verbose option is in effect, inode numbers are also listed.

`cd` *directory*
Changes to directory *directory* (within the dump tape).

`pwd` Prints the full path name of the current working directory.

`add` [*filename*]
Adds the current directory, or the named file or directory *directory* to the list of files to extract. If a directory is specified, add that directory and its files (recursively) to the extraction list (unless the `h` modifier is in effect).

`delete` [*filename*]
Deletes the current directory, or the named file or directory from the list of files to extract. If a directory is specified, delete that directory and all its descendents from the extraction list (unless the `h` modifier is in effect). The most expedient way to extract a majority of files from a directory is to add that directory to the extraction list, and then delete specific files to omit.

`extract`
Extracts all files on the extraction list from the dump tape. `vxrestore` asks which volume the user wishes to mount. The fastest way to extract a small number of files is to start with the last tape volume and work toward the first. The owner, modification time, and mode are restored (if possible).

`verbose`
Toggles the status of the `v` modifier. While `v` is in effect, the `ls` command lists the inode numbers of all entries, and `vxrestore` displays information about each file as it is extracted.

`help` Displays a summary of the available commands.

`quit` `vxrestore` exits immediately, even if the extraction list is not empty.

NOTES

`vxrestore` can get confused when doing incremental restores from dump tapes that were made on active file systems. Dumps should be made using the snapshot mount facilities of VxFS.

A "level `0`" dump must be done after a full restore. Because `vxrestore` runs in user mode, it has no control over inode allocation; this means that `vxrestore` repositions the files, although it does not change their contents. Thus, a full dump must be done to get a new set of directories reflecting the new file positions, so

that later incremental dumps will be correct.

DIAGNOSTICS

`vxrestore` complains about bad option characters.

Read errors result in complaints. If `-y` has been specified, or the user responds `y`, `vxrestore` will attempt to continue.

If the dump extends over more than one tape, `vxrestore` asks the user to change tapes. If the `-x` or `-i` option has been specified, `vxrestore` also asks which volume the user wishes to mount.

There are numerous consistency checks that can be listed by `vxrestore`. Most checks are self-explanatory or can "never happen". Common errors are given below.

`Converting to new file system format.`
: A dump tape created from the old file system has been loaded. It is automatically converted to the new file system format.

filename`: not found on tape`
: The specified file name was listed in the tape directory, but was not found on the tape. This is caused by tape read errors while looking for the file, and from using a dump tape created on an active file system.

`expected next file` *inumber*`, got` *inumber*
: A file that was not listed in the directory showed up. This can occur when using a dump tape created on an active file system.

`Incremental tape too low`
: When doing an incremental restore, a tape that was written before the previous incremental tape, or that has too low an incremental level has been loaded.

`Incremental tape too high`
: When doing incremental restore, a tape that does not begin its coverage where the previous incremental tape left off, or one that has too high an incremental level has been loaded.

`Tape read error while restoring` *filename*
`Tape read error while skipping over inode` *inumber*
`Tape read error while trying to resynchronize`
`A tape read error has occurred.`
: If a file name is specified, then its contents are probably partially wrong. If an inode is being skipped or the tape is trying to resynchronize, then no extracted files have been corrupted, though files may not be found on the tape.

`resync vxrestore, skipped` *num*
: After a tape read error, `vxrestore` may have to resynchronize itself. This message lists the number of blocks that were skipped over.

FILES

`/dev/rmt8`
: the default tape drive

`/tmp/rstdir*`
: file containing directories on the tape

`/tmp/rstmode*`
: owner, mode, and timestamps for directories

`./restoresymtable`
: information passed between incremental restores

SEE ALSO

`vxdump`(1M), `mkfs`(1M), `mount`(1M).

System Administrator's Reference Manual

wall(1M)

NAME

`wall -` write to all users

SYNOPSIS

```
/usr/sbin/wall
```

DESCRIPTION

`wall` reads its standard input until an end-of-file. It then sends this message to all currently logged-in users preceded by:

```
Broadcast Message from ...
```

It is used to warn all users, typically prior to shutting down the system.

The sender must have superuser rights to override any protections the users may have invoked [see `mesg(1)`].

`wall` runs `setgid()` [see `setuid(2)`] to the group ID `tty`, in order to have write permissions on other user's terminals.

`wall` will detect non-printable characters before sending them to the user's terminal. Control characters will appear as a ^ followed by the appropriate ASCII character; characters with the high-order bit set will appear in meta notation. For example, `\003` is displayed as `^C` and `\372` as `M-z`.

LOCALE

The `LC_MESSAGES` environment variable governs the language in which message texts are displayed.

If `LC_MESSAGES` is undefined or is defined as the null string, it defaults to the value of `LANG`. If `LANG` is likewise undefined or null, the system acts as if it were not internationalized.

If any of the locale variables has an invalid value, the system acts as if none of the variables was set.

The `LC_ALL` environment variable governs the entire locale. `LC_ALL` takes precedence over all the other environment variables that affect internationalization.

DIAGNOSTICS

"`Cannot send to ...`" when the open on a user's tty file fails.

FILES

```
/dev/tty*
```

SEE ALSO

`mesg(1)`, `write(1)`.

System Administrator's Reference Manual

whodo(1M)

NAME

whodo - who is doing what

SYNOPSIS

/usr/sbin/whodo [–h] [–l] [*user*]

DESCRIPTION

whodo produces formatted and dated output from information in the /var/adm/utmp, /etc/ps_data, and /proc/pid files.

The display is headed by the date, time, and machine name. For each user logged in, device name, login-ID, and login time is shown, followed by a list of active processes associated with the login-ID. The list includes the device name, process-ID, CPU minutes and seconds used, and process name.

If *user* is specified, output is restricted to all sessions pertaining to that user.

OPTIONS

–h Suppresses the heading.

–l Produces a long form of output. The fields displayed are: the user's login name, the name of the tty the user is on, the time of day the user logged in (in *hours*:*minutes*), the idle time — that is, the time since the user last typed anything (in *hours*:*minutes*), the CPU time used by all processes and their children on that terminal (in *minutes*:*seconds*), the CPU time used by the currently active processes (in *minutes*:*seconds*), and the name and arguments of the current process.

EXAMPLES

The command:

whodo

produces a display like this:

```
Tue Mar 12 15:48:03 1993
SINIX-P
tty09    hofu      8:51
    tty09   28158    0:29 ksh
tty52    jjb      15:23
    tty52   21688    0:05 sh
    tty52   22788    0:01 whodo
    tty52   22017    0:03 vi
    tty52   22549    0:01 sh
xt162    kneba    10:20
    tty08    6748    0:01 layers
    xt162    6751    0:01 sh
    xt163    6761    0:05 sh
    tty08    6536    0:05 sh
```

DIAGNOSTICS

If the PROC driver is not installed or configured or if /proc is not mounted, a message to that effect is issued and whodo will fail.

The exit status is zero on success, non-zero on failure.

FILES

/etc/passwd

/etc/ps_data

```
/var/adm/utmp
/proc/pid
```

SEE ALSO

ps(1), who(1).

System Administrator's Reference Manual

wsa(1M)

NAME

`wsa` - start the WebSysAdmin GUI component as an application

SYNOPSIS

`/opt/wsa/bin/wsa` [`-r`] [`-l` *lang*] *server*

DESCRIPTION

`wsa` starts the GUI component (Graphical User Interface) of WebSysAdmin as an application.

`wsa` must be called to use WebSysAdmin on a Solaris 2.x PC or on the LAN console or to run the Java Runtime Environment (JRE) on RM200/RM300.

`wsa` need not be called to use Netscape from Version 4.x under Windows 95 or NT4.0.

The default port number for a UDP connection is 8880.

OPTIONS

`-r` WebSysAdmin is only started in read mode. If this option is not specified, WebSysAdmin will be started in write mode.

`-l` *lang*

WebSysAdmin is started with the specified language setting *lang*. *lang* can be one of the two-position language codes from the ISO 639 norm. The languages supported in the WebSysAdmin basic package are `en` (English) and `de` (German). The default is English (`-l en`).

For other languages, the relevant language catalog must be installed for WebSysAdmin.

server

Server to which WebSysAdmin is connected over the LAN. The symbolic name can be specified or the name entered in Internet dot notation.

NOTES

The Java Runtime Environment (JRE) must be installed for WSA on the local machine (version ≥ jre1.2.1).

The `SMAWjvrt` package must be installed from the SYS-SOL CD when using a Solaris PC.

The `install_wsa_gui` script is provided on the Reliant UNIX CD for installing the GUI component on the local host.

The Reliant UNIX CD can be accessed locally or remotely via NFS.

Examples:

a) remote:

Mount the Reliant UNIX CD in RM server.

```
server:  mount /cdrom
server:  share -F nfs -o root=localpc,ro=localpc /cdrom

localpc: mount -F nfs server:/cdrom  /cdrom/cdrom0
localpc: /cdrom/cdrom0/install_wsa_gui
```

b) local:

Mount Reliant UNIX CD on the local host (e.g. Solaris 2.x PC). `vold` mounts the CD automatically under its VOLUME name. If `vold` is not running, the CD can also be mounted manually:

```
mount -F hsfs -o ro /dev/dsk/c1t0d0p0 mountpoint

localpc: /cdrom/reliantunix_y_v5_45a1020/install_wsa_gui
```

If the local access path to the CD does not begin with `/cdrom`, this access path must be passed in the call:

```
localpc: /cdrom/reliantunix_y_v5_45a1020/install_wsa_gui mountpoint
```

Following successful execution of `install_wsa_gui`, `wsa` is available under `/opt/wsa/bin`; it does make sense, therefore, to extend the `PATH` variable by the `/opt/wsa/bin` path.

System Administrator's Reference Manual

xbload(1M)

NAME

`xbload, load -` load a workstation program onto a BA80 orTC20-V105workstation

SYNOPSIS

`xbload` [`–adl`] [`–u`] [`–p` *file*] [`–f` *file*] *terminal_path* [*ttydefs_label*]

`xbload –?`

DESCRIPTION

`xbload` checks the load state of the specified terminal. The terminal is set up if it has not been already set up, or if the `–l` option is used. If the TC20-V105 can be loaded, the variant and version of the firmware are checked, and reloaded as required. There are different variants of the TC20-V105 firmware (with varying functionality). There are also different versions (development status) of each variant.

`xbload` first sets the line parameters for the terminal being set up. The basic and default setting for the physical line parameters is `9600`, `cs8`, `cread`. The argument *ttydefs_label* can be used to change this setting by placing a tag in `/etc/ttydefs`.

The following specifications are recognized and supported:

- Baud rate
- Character length (CS8, CS7 etc.)
- Parity
- Stop bits.

A request is then made to the terminal to identify itself. If necessary, this request is repeated 5 times. If the program does not need to be loaded, it is terminated with the return value 0. Otherwise, the workstation program is loaded. The load state of the terminal is then checked again, and the loading procedure is repeated up to 5 times, if required.

OPTIONS

`–a` The software is transferred in ASCII format (INTELHEX).

Note:

In this case, `DATA FORMAT INTEL` must be entered in the BA80 load menu. Otherwise, the data is transferred in binary format, which requires the setting `DATA FORMAT HEX`. This is of no significance for TC20-V105.

`–d` This option activates debug mode for outputting trace information (usually to `stderr`). `–d` sets the debug level to 1, `–dd` to 2 and so on.

`–l` Unconditional setup. Before the terminal is identified/set up, it is requested by means of an ESC sequence to go to the load Prom. This option can also be used to set up terminals that have already been set up.

`–p` *file*

A terminal that has not been set up (or if the `–l` option was used) is set up with the specified software. The firmware for a TC20-V105 is compared with the firmware under this file name, and if necessary the TC20-V105 is reloaded (see `–u` option).

If the option is not specified:

Where a BA80 has not been set up, then when there is a load request, a file name is taken from `/opt/xb` so that it can be set up. In the case of a TC20, the file name under which a firmware is expected on the host side, is determined from the variant *VAR* of the terminal: `/etc/tc20/`*VAR*`.default`.

`–f` *file*

All output (error messages, debug output) is normally sent to `stderr`. With this option, the output is rerouted to the specified file.

`–u` Weak update: In the case of a TC20-V105 that has been set up, a comparison is made of the version of firmware loaded and the firmware on the host side. (The firmware on the host is specified either with `–p`*file* or is taken from `/etc/tc20/`*VAR*`.default`, where *VAR* is the variant of the TC20-V105 set up.) If this comparison establishes that with both variants the firmware on the host is a more recent version (higher version), the TC20-V105 is reloaded with the more recent version (otherwise, it is not). If it is established that both the loaded firmware and that on the host are different in the variant, an error message is output.

`–uu` Strong update (see `–u` option): The TC20-V105 is set up again if the loaded firmware and that on the host differ in variant or version. This option must be used if, in conjunction with the `–p` *file* option, a TC20-V105 is to be reloaded with another variant.

terminal_path
The device name of the terminal to be loaded in the file system. A TTY interface is expected.

ttydefs_label
With this option, the `/etc/ttydefs` file is tagged in order to define the TTY specification. The physical line parameters are set in accordance with the specifications of this tag. If this option is not specified, `9600`, `cs8`, `cread` is used. If the pseudo tag `none` is specified instead of a tag, the physical parameters remain unchanged.

`–?` The command syntax is displayed on `stderr`.

NOTE

Two different `xb` workstation program formats are supported: In addition to the conventional workstation programs with the format ID 01, which contain the entire load software in one file, there are also programs with the format ID 02 (extended format). These contain the names of the files, specified in the order in which they are to be loaded, and have the following format:

`02;`*filename1*`[;`*filename2*`] ... # Comment`

HDLC-BA80 can only be loaded in binary format, i.e. without the `–a` option.

DIAGNOSTICS

The program ends with the return value 0, if the terminal was already set up or has been set up successfully. If any errors occur, the return values have the following meaning:

`1` = Syntax error. A command syntax is displayed.

`2` = Signal (`SIGTERM`, `SIGHUP`, `SIGINT`) received.

`3` = Error in input/output operations such as `open`, `read`, `write`, `ioctl` and `fcntl`.

`4` = Program aborted after several unsuccessful load attempts.

`5` = Unknown or no response from the TTY during identification. This may be the result of inconsistencies in the line parameters, or the fact that the terminal is not a BA80/9766 or TC20.

`6` = The desired workstation program was not found. This error code also appears if the workstation program has an unknown format ID or (in the case of extended format) a specified file from this program could not be opened.

`10` =
ATC20 has loaded old or unsuitable firmware, but the options (`–u`/`–uu`) do not permit a reload. The TC20-V105 could not be put in load mode and loading was terminated by the TC20.

Error messages and trace information (provided the `–d` option is set) are output to `stderr`; they can be rerouted using the `–f` option. Successful loading and errors which occur when loading TC20-V105 are logged in `/var/tc20/load_log` and also in Logging 3.0 under the component number 1080011.

FILES

`/etc/ttydefs`

```
/opt/xb
/etc/tc20
/var/tc20/load_log
```

System Administrator's Reference Manual

xscon(1M)

NAME

`xscon` - map consoles

SYNOPSIS

`xscon` [*systemname1*, *systemname2* ...]

DESCRIPTION

`xscon` enables consoles from other systems to be mapped to an X-terminal. All server consoles to which a connection can be established are offered for selection in a check list. By clicking on the "Establish connections" button, a terminal emulation is started for each selected server console. In this terminal emulation, the connection from an RM system to a server console is established using the `cu`(1) command. Either `xpct`, `xterm` or `x97801` can be selected from a check list as the terminal emulation to be started. `xpct` is the default selection in this check list. You can override this preselection using an environment variable: `XSCON_EMULATION=xterm` or `XSCON_EMULATION=x97801`.

If the `SItelesv` package is installed, buttons will be available to enable or disable teleservice operating. If `enable` is selected, all connections are started with teleservice operating, that means with a `telepty` process. The default setting of the button `Teleservice Operating:` can be controlled by the resource `xscon*teleservice: false` (default) means `disable` is preselected, otherwise use `xscon*teleservice: true`. If this button is not to appear, even if the `SItelesv` package is installed, the resource `xscon*noteleservice: true` can be used.

`xscon` is terminated immediately once you click on the "Establish connections" button.

When starting `xscon`, you can also specify the names of the systems to which a connection is to be established as call parameters. In this case, the `xscon` application window is not displayed.

If the connection to a server console cannot be established (e.g. because it is already being used by another terminal), the corresponding `cu` error message is displayed in the emulation window. In this case, the window is closed automatically after 10 seconds. If the "xpct" terminal emulation is selected, the mechanism provided by this window for generating a log file can be used. This mechanism is started from the `xpct` control menu. The name of the log file is preset: `/tmp/XSCON.`*systemname*. This log file is stored on the system from which `xscon` was started.

If you want to operate terminal-dependent applications (e.g. `sysadm`) on a console from a terminal emulation, the `TERM` environment variable should be set for the console according to the appropriate terminal emulation:

Terminal emulation	**TERM** variable
`xterm`	`xterm`
`x97801`	`97801`
`xpct`	`ansi-tc20-c`

The variable can be set interactively after you log on to the console, or by entering it in the `/etc/ttytype` file using an editor, or by modifying this entry for the console (should it already exist).

Requirements for starting xscon

You must first of all configure the terminal lines, over which `cu` is to establish the connections to the consoles, on the RM system. This is done using `sysadm` or the `termadd`(1M) command. Please note the following when configuring these terminal lines:

- The terminal lines are to be configured "without modem control", i.e. the port number begins with the character `n`.
- A login process must not be activated on the line to be configured. This can be done by using the `-d` option (deactivation flag) with the `termadd` command, or by configuring the line as a serial line under

sysadm.

After configuring the terminal lines, the owner of these terminal lines has to be changed to uucp by using the chown(1) command.

The corresponding uucp files /etc/uucp/Devices and /etc/uucp/Systems must then be administered. Consoles to be displayed externally with xscon must be entered in these files with the alias Scon*X* (*X* = 0...*n*). This can be done using sysadm, or the entries can be incorporated using a text editor as in the example below.

Example:

Entries in the /etc/uucp/Devices file:

```
Scon0 term/0 - 9600 direct
Scon1 term/1 - 19200 direct
Scon2 term/2 - 9600 direct
Scon3 term/3 - 9600 direct
```

Entries in the /etc/uucp/Systems file:

```
uptrend Any     Scon0 9600
usher   Any     Scon1 19200
upwind  Any     Scon2 9600
useful  Any     Scon3 9600
```

xscon only can handle up to 20 configurations.

RESOURCES

You can configure the emulation windows to be started using the usual resources.

This xterm window's instance identifier is xscon and its class identifier is XPct or XTerm.

FILES

/etc/uucp/Devices

/etc/uucp/Systems

SEE ALSO

cu(1) , termadd(1M) , xpct(1X), xterm(1X).

System Administrator's Reference Manual

zdump(1M)

NAME

`zdump` - time zone dumper

SYNOPSIS

`zdump` [−v] [−c *cutoffyear*] [*zonename* ...]

DESCRIPTION

The `zdump` command prints the current time in each *zonename* named on the command line.

OPTIONS

−v
: For each *zonename* on the command line, prints the current time, the time at the lowest possible time value, the time one day after the lowest possible time value, the times both one second before and exactly at each time at which the rules for computing local time change, the time at the highest possible time value, and the time at one day less than the highest possible time value. Each line ends with `isdst=1` if the given time is Daylight Saving Time or `isdst=0` otherwise.

−c *cutoffyear*
: Cuts off verbose output near the start of the year *cutoffyear*.

LOCALE

The `LC_MESSAGES` environment variable governs the language in which message texts are displayed.

The `LC_TIME` environment variable governs the format of date and time strings.

If `LC_MESSAGES` or `LC_TIME` is undefined or is defined as the null string, it defaults to the value of `LANG`. If `LANG` is likewise undefined or null, the system acts as if it were not internationalized.

If any of the locale variables has an invalid value, the system acts as if none of the variables was set.

The `LC_ALL` environment variable governs the entire locale. `LC_ALL` takes precedence over all the other environment variables that affect internationalization.

FILES

`/usr/lib/locale/TZ`
: standard zone information directory

SEE ALSO

`zic`(1M), `ctime`(3C).

System Administrator's Reference Manual

zic(1M)

NAME

zic - time zone compiler

SYNOPSIS

zic [–s] [–v] [–d *directory*] [–l *timezone*] [*filename ...*]

DESCRIPTION

zic reads text from the file(s) named on the command line and creates the time conversion information files specified in this input. If a *filename* is –, standard input is read.

Input lines are made up of fields. Fields are separated by any number of white space characters. Leading and trailing white space on input lines is ignored. A pound sign (#) in the input introduces a comment which extends to the end of the line on which the pound sign appears. White space characters and pound signs may be enclosed in double quotes (") if they're to be used as part of a field. Any line that is blank (after comment stripping) is ignored. Non-blank lines are expected to be of one of three types: rule lines, zone lines, and link lines.

A rule line has the form

Rule	NAME	FROM	TO	TYPE	IN	ON	AT	SAVE	LETTER/S

For example:

```
Rule      USA    1969  1973   –    Apr lastSun 2:00  5:00      D
```

The fields that make up a rule line are:

NAME Gives the (arbitrary) name of the set of rules this rule is part of.

FROM Gives the first year in which the rule applies. The word minimum (or an abbreviation) means the minimum year with a representable time value. The word maximum (or an abbreviation) means the maximum year with a representable time value.

TO Gives the final year in which the rule applies. In addition to minimum and maximum (as above), the word only (or an abbreviation) may be used to repeat the value of the FROM field.

TYPE Gives the type of year in which the rule applies. If TYPE is – then the rule applies in all years between FROM and TO inclusive; if TYPE is uspres, the rule applies in U.S. Presidential election years; if TYPE is nonpres, the rule applies in years other than U.S. Presidential election years. If TYPE is something else, then zic executes the command

yearistype *year type*

to check the type of a year: an exit status of zero is taken to mean that the year is of the given type; an exit status of one is taken to mean that the year is not of the given type.

IN Names the month in which the rule takes effect. Month names may be abbreviated.

ON Gives the day on which the rule takes effect. Recognized forms include:

5 the fifth of the month

lastSun
the last Sunday in the month

lastMon
the last Monday in the month

Sun>=8
first Sunday on or after the eighth

Sun<=25
last Sunday on or before the 25th

Names of days of the week may be abbreviated or spelled out in full. Note: there must be no spaces within the `ON` field.

`AT` Gives the time of day at which the rule takes effect. Recognized forms include:

`2` time in hours

`2:00` time in hours and minutes

`15:00` 24-hour format time (for times after noon)

`1:28:14`
time in hours, minutes, and seconds

Any of these forms may be followed by the letter `w` if the given time is local "wall clock" time or `s` if the given time is local "standard" time; in the absence of `w` or `s`, wall clock time is assumed.

`SAVE` Gives the amount of time to be added to local standard time when the rule is in effect. This field has the same format as the `AT` field (although, of course, the `w` and `s` suffixes are not used).

`LETTER/S`
Gives the "variable part" (for example, the "S" or "D" in "EST" or "EDT") of time zone abbreviations to be used when this rule is in effect. If this field is `–`, the variable part is null.

A zone line has the form

Zone NAME GMTOFF RULES/SAVE FORMAT [UNTIL]

For example:

```
Zone    Australia/South    9:00        Oz          CST
```

The fields that make up a zone line are:

`NAME` The name of the time zone. This is the name used in creating the time conversion information file for the zone.

`GMTOFF`
The amount of time to add to GMT to get standard time in this zone. This field has the same format as the `AT` and `SAVE` fields of rule lines; begin the field with a minus sign if time must be subtracted from GMT.

`RULES/SAVE`
The name of the rule(s) that apply in the time zone or, alternately, an amount of time to add to local standard time. If this field is – then standard time always applies in the time zone.

`FORMAT`
The format for time zone abbreviations in this time zone. The pair of characters `%s` is used to show where the "variable part" of the time zone abbreviation goes.

`UNTIL` The time at which the GMT offset or the rule(s) change for a location. It is specified as a year, a month, a day, and a time of day. If this is specified, the time zone information is generated from the given GMT offset and rule change until the time specified.

The next line must be a "continuation" line; this has the same form as a zone line except that the string "Zone" and the name are omitted, as the continuation line will place information starting at the time specified as the `UNTIL` field in the previous line in the file used by the previous line. Continuation lines may contain an `UNTIL` field, just as zone lines do, indicating that the next line is a further continuation.

A link line has the form

`Link` *LINK-FROM* LINK-TO

For example:

`Link` US/Eastern EST5EDT

The *LINK-FROM* field should appear as the `NAME` field in some zone line; the *LINK-TO* field is used as an

alternate name for that zone.

Except for continuation lines, lines may appear in any order in the input.

OPTIONS

`-s` (short) The rule valid since 1970 applies if the word `minimum` appears in the field *FROM*.

`-v` Complains if a year that appears in a data file is outside the range of years representable by system time values (0:00:00 AM GMT, January 1, 1970, to 3:14:07 AM GMT, January 19, 2038).

`-d` *directory*
Creates time conversion information files in the directory *directory* rather than in the standard directory `/usr/lib/locale/TZ`.

`-l` *timezone*
Uses the time zone *timezone* as local time. `zic` will act as if the file contained a link line of the form

`Link` *timezone localtime*

LOCALE

The `LC_MESSAGES` environment variable governs the language in which message texts are displayed.

If `LC_MESSAGES` is undefined or is defined as the null string, it defaults to the value of `LANG`. If `LANG` is likewise undefined or null, the system acts as if it were not internationalized.

If any of the locale variables has an invalid value, the system acts as if none of the variables was set.

The `LC_ALL` environment variable governs the entire locale. `LC_ALL` takes precedence over all the other environment variables that affect internationalization.

NOTE

For areas with more than two types of local time, you may need to use local standard time in the `AT` field of the earliest transition time's rule to ensure that the earliest transition time recorded in the compiled file is correct.

FILES

`/usr/lib/locale/TZ`
standard directory used for created files

SEE ALSO

`sh(1)`, `time(1)`, `zdump(1M)`, `ctime(3C)`, `timezone(3C)`, `timezone(4)`.

File Formats(4)

Introduction

System Administrator's Reference Manual

intro_sarm4(4)

NAME

`intro_sarm4 -` introduction to file formats

DESCRIPTION

This section outlines the formats of various files. The C structure declarations for the file formats are given where applicable. Usually, the header files containing these structure declarations can be found in the directories `/usr/include` or `/usr/include/sys`. For inclusion in C language programs, however, the syntax `#include <`*filename*`.h>` or `#include <sys/`*filename*`.h>` should be used.

_upstab(4) - cron(4)

System Administrator's Reference Manual

_upstab(4)
RM600 only

NAME

_upstab - parameterization file for the ubft daemon

SYNOPSIS

/var/adm/ubft/_upstab

DESCRIPTION

Reactions of the ubft daemon to error events are parameterized, and settings to reflect the hardware configuration are made by means of entries in the _upstab file. The following items can be parameterized:

- Acknowledgement of the UPS/BBU configuration by the CONFIG interface (see below).
- Specification of the UPS interface type (only for connecting a UPS to a system cabinet with a CSI as boot controller, SY54 or SY56).
- Safe startup (see below).
- Configuration of error events: The delay time, user reaction script time, warning time, reaction, as well as optional user reaction script parameters are set for each cabinet, and for each event.
- The period of time after which the time of the EIP is to be synchronized periodically with that of the CSI (only for systems with CSI as boot controller, i.□e. system cabinet SY54 or SY56).
- Allocation of the converters of the external BBU to the cabinets they supply.
- Allocation of the UPS to the cabinets it supplies.
- Configuration of information events: An optional user reaction script parameter is set for each cabinet and for each event.

If no _upstab file exists when the ubft daemon is started, the ubft daemon creates a file containing default values.

The default values specified below for individual parameters are also valid if there is no definition line for this parameter in the _upstab file when the ubft daemon is started.

The following line indicates whether or not a UPS/BBU configuration is taken into account by the CONFIG interface:

hw_config:*value*

where *value*::= { yes | no }

Example:

hw_config:yes

The default value is yes.

If yes is set, it is expected that the configuration of the UPS interface type (ups_type:...), the BBU converter cabinet allocation (bbu:...), and the UPS cabinet allocation (ups:...) will be handled by the CONFIG interface. In this case, the corresponding entries in the _upstab file are not considered. If no is set, the _upstab configuration is used.

If safe startup is set, the system administrator is asked at the time of startup whether startup should continue despite uncharged UPS or BBU batteries. If the system administrator answers no, startup in administrator mode (run level 1) or multiuser mode (run level 2) is awaited until the UPS or BBU reports that the batteries have been loaded. The question is asked again if the system administrator interrupts the wait procedure with q<CR>. Safe startup is only valid for the UPS and BBU types that provide information on the load status of the batteries.

Safe startup is defined by the following line:

safe_boot:*value*

where *value*::= { `yes` | `no` }

Example:

```
safe_boot:no
```

The default value is `no`.

In the case of systems with a UPS connection to a system cabinet with CSI as boot controller (SY54 or SY56), the UPS interface type is defined by a line with the following format:

`ups_type:`*UPS_type*

where *UPS_type*::= { `unused` | `9071` | `B40` }

Example:

```
ups_type:unused
```

The default value is `unused`.

The value of this parameter is written to the NVRAM of the CSI and evaluated by the CSI.

The UPS B41 allows use of either a UPS 9071 or a UPS B40 interface. The interface type must then be set accordingly to `9071` or `B40`.

In the case of systems with EIP3 master, a corresponding configuration line is ignored because the UPS is connected via the EIP3, which recognizes the UPS interface type automatically.

A line in the following format determines the number of seconds after which the CSI periodically synchronizes the clock of the master EIP on the system with its own clock, in order to synchronize the times of all EIPs on the system:

`sync_csi/eip:`*secs*

where *secs* is a positive integer value or 0.

Example:

```
sync_csi/eip:86400
```

Synchronization is only performed on systems with CSI as the boot controller. In the case of systems with EIP3 master (HIOS/EHIOS as the boot controller) a corresponding configuration line is ignored. The following applies for the *secs* value:

- The unit of the *secs* value is seconds.
- If *secs* has the value 0, synchronization does not take place.
- If the *secs* value lies in the interval [1,9], the value 10 is used.
- The standard value is 7,200 seconds, i.e. two hours.

Reaction type parameters follow line by line. Reaction type parameters allow parameterization of the reaction model used when error events occur. Important parameters include the period of time between recognizing the error and starting the shutdown action, as well as the type of shutdown action. The reaction model also allows you to call up a specific user reaction script `etc/opt/havail/user_reaction`, and – if one exists – to suppress the shutdown action if necessary. The reaction type parameters have the following format:

error:*cabno*:*waittime*:*scripttime*:*warntime*:*reaction*[:*item*]

The error event is identified with *error*. Values can be taken from the following list.:

```
error ::= { ups_active | acf2 | fan_unit_fail
          | temp_fail | acf2+extbbu_active | acf2+intbbu_active
          | acf2+combbbu | ups_active+defect | ups_active+batlow | acf1
          | power_fan_fail | acf2+extbbu_end | acf2+extbbu_service
          | acf2+ups/extbbu_notfull | acf2+extbbu_defect
          | too_few_ps_units
        }
```

The *cabno* parameter contains the cabinet number or the character string `any`. In the case of systems with EIP2 master, the cabinet number corresponds to the logical cabinet ID from the range of values [0,15]. In the case of systems with EIP3 master, the cabinet number corresponds to the physical cabinet ID from the range

of values [0,63]. The definition line with the character string `any` is used for any cabinets in the system that are not defined explicitly by their own line and cabinet number.

Where there are several entries with the same error event *error* and the same cabinet number *cabno*, only the first is taken.

The *waittime* parameter describes the delay time in seconds, before the user reaction script `user_reaction` is called for the first time. The *scripttime* parameter describes the time that is available to the user reaction script to execute its actions. If a warning time *warntime* greater than zero is specified, the user reaction script *warntime* is called up again seconds before the shutdown action is initiated.

The sum of the delay time, user reaction script time, and warning time represents the period of time from when the error is recognized to when the corresponding shutdown action is started.

The following information should be noted with regard to the definition of times: The value 0 can also be defined for *scripttime* and *warntime*. In this case, a reaction is not executed for every stage of the reaction model. If *scripttime*=0, the user reaction script `user_reaction` is **not** called, regardless of the other times.

A warning time is really only effective then if *scripttime* is greater than 0.

The type of shutdown action is defined by the *reaction* parameter. Possible values include the following:

reaction ::= { `no_shutdown` | `slow_shutdown` | `fast_shutdown` }

With `slow_shutdown`, "shutdown -y -g0 -i0" is executed, and with `fast_shutdown` "uadmin 2 0" is executed. If `no_shutdown` is defined, no shutdown action is executed, and no user reaction script `user_reaction` is called.

The *item* parameter allows the option of specifying a character string. The character string begins after the colon with the first character, other than a blank or a tab, and ends with the last character before the line end, which is also not a blank or a tab. The character string must not contain colons. If a character string has been specified, it is included as the final parameter every time the user reaction script `user_reaction` is called. It is possible to distinguish the error events in the user reaction script by specifying unique character strings for all error events.

Information event parameters follow line by line, which can be used to configure the calling of the user reaction script `user_reaction`. The information event parameters have the following format:

info:*cabno*:*flag*[:*item*]

The information event is identified with *info*. Values can be taken from the following list:

info ::= { `cab_lost` | `cab_found` | `cab_on` | `cab_off`}

The *cabno* parameter contains the cabinet number from the range of values [0,15] for systems with EIP2 master, or from the range of values [0,63] for systems with EIP3 master or the character string `any` (see above).

The `cab_on` and `cab_off` information events are only generated in systems with EIP3 master. In the case of systems with EIP2 master, the `cab_found` and `cab_lost` events are generated instead.

The *flag*::= { `yes` | `no` } parameter is used to specify whether or not the user reaction script is called when the information event occurs. The default value is `no`.

If `yes` is specified, and an information event occurs for a cabinet *cabno*, the user reaction script `user_reaction` is called once with the parameters `INFO`, `OCCURRED`, the cabinet number *cabno*, the number `0`, and the character string *item* (optional). The structure of the character string *item* is identical to that of the error event (see above).

The `_upstab` file can contain a number of lines which define the allocation of the BBUs converters to the cabinets supplied. These lines have the following format and begin with the string `bbu:`:

`bbu:`*bbucabno*:*converterno*:*cabno* [*cabno* ...]

The *bbucabno* parameter contains the cabinet number of the external BBU cabinet. The *converterno* parameter specifies the number of the converter from the range of values [0,5]. This is followed by a list of the numbers

of the cabinets, separated by blanks, which are supplied by the specified converter. In the case of the *cabno* and *bbucabno* parameters, the logical cabinet IDs are to be specified for systems with EIP2 master from the range of values [0,15], and the physical cabinet IDs for systems with EIP3 master from the range of values [0,63].

If no BBU converter cabinet allocation is specified (default situation), the application program is invoked with the cabinet number of the BBU cabinet, if the BBU converter is active (power failure).

If an active BBU converter was only assigned numbers of non-existent cabinets, the application program is not invoked.

The `_upstab` file can contain several lines which define the allocation of UPSs to the supplied cabinets. These lines have the following format:

`ups`*upscabno* `:` *cablist*

The *upscabno* parameter contains the number of the cabinet to which the signal interface of the UPS is connected. If this entry is missing, it is assumed that the UPS is connected to the system cabinet.

The *cablist* parameter either contains the value `any`, if the UPS supplies all cabinets in the system, or a list of numbers of cabinets supplied by UPS, separated by blanks:

cabno [*cabno* ...]

In the case of systems with EIP2 master, the logical cabinet IDs from the range of values [0,15] are specified for *cabno* and *upscabno*, whereas in the case of systems with EIP3 master, the physical cabinet IDs from the range of values [0,63] are given.

If no UPS cabinet allocation is specified (default situation), the application program is invoked with the cabinet number *upscabno* in the case of an active U|PSV (power failure).

If an active UPS was only assigned numbers of non-existent cabinets, the application program is not invoked.

Unrecognizable entries in the `_upstab` file are ignored, and logged in the log file of the `ubft` daemon.

NOTES

The UPS interface type, the BBU converter cabinet allocation, and the UPS cabinet allocation cannot be determined automatically. For the system to react correctly in the case of a power failure, corresponding entries must be made by a system administrator in the `_upstab` file or using the CONFIG interface.

The `_upstab` file is evaluated when the `ubft` daemon is started. If changes to the configuration are made during operation, the `ubft` daemon must be prompted with `kill -2` *ups_pid* to evaluate the `_upstab` file again. The process number of the `ubft` daemon must be specified for *ups_pid*.

If `hw_config:yes` is set, the `ubft` daemon initiates creation of the `/var/adm/ubft/hwc_upstab` file every time the `_upstab` file is evaluated. This `/var/adm/ubft/hwc_upstab` file has the same structure as the `_upstab` file, but only contains the configuration data from the HW_CONFIG configuration. The configuration data from the `_upstab` file is then redefined by the configuration data in the `hwc_upstab` file. If the configuration data in the `hwc_upstab` file is incomplete, the corresponding entries in the `_upstab` file are used.

Comment lines in the `_upstab` file begin with the # character, but can be preceded by blanks or tabs. Comment lines are likewise ignored. Comment must **not** be specified in a definition line from after the definition to the end of the line.

If the `_upstab` file is generated by the `ubft` daemon, the `ubft` daemon version is specified as comment in the first line (e.g. `# VERSION=3.00`).

FILES

`/var/adm/ubft/hwc_upstab`
: HW_CONFIG parameterization file for UPS/BBU configuration

`/etc/opt/havail/user_reaction`
: User reaction script for shutdown events

SEE ALSO

ups(1M), ubft(7), user_reaction(7).

System Administrator's Reference Manual

_upstab(4)
RM400 only

NAME

_upstab - parameterization file for the eipd, ptfd, andpupsd daemon

SYNOPSIS

/var/adm/ubft/_upstab

DESCRIPTION

Reactions of the eipd, eipd, ptfd, and pupsd daemon to error events can be parameterized by means of entries in the _upstab file.

Reaction type parameters allow parameterization of the reaction model used during error events. Important parameters are the timespan between detection of the error and startup of the shutdown action, and the type of shutdown action. The reaction type parameters have the following format:

error:*cabno*:*waittime*:*scripttime*:*warntime*:*reaction*:*script* [*parameter*]

The error event is named using *error*. Values can be taken from the following list:

error ::= {acf2 | fan_unit_fail | temp_fail | acf1 | eip_lost | too_few_ps_units | \ups_active | ups_active+batlow}

The *cabno* parameter contains the logical cabinet ID (logical cabinet number) from the range of values [0,15] or the character string any. The definition line with the character string any is used for those cabinets in the system which are not explicitly defined by an individual line and cabinet number.

In the case of multiple entries with the same error event *error* and the same cabinet number *cabno*, only the first is taken into account.

The *scripttime* parameter describes the user reaction script time available to the user to execute the required actions.

Warning:

waittime and *warntime* have not yet been evaluated on the RM400.

The sum of the delay time and user reaction script time represents the timespan between detection of the error event and startup of the relevant shutdown action.

The *reaction* parameter defines the type of shutdown action. Possible values are:

reaction ::= { no_shutdown | slow_shutdown | fast_shutdown }

The parameter "script [parameter]" can be used to start a user-specific program before execution of the reaction. "script [parameter]" is first supplied with "echo".

If slow_shutdown is used, "shutdown –y –g*warntime* –i0" is executed, and if fast_shutdown is used, "uadmin 2□0" is executed. If no_shutdown is defined, no shutdown action is executed, but the "user_reaction" program is executed.

NOTES

The _upstab file is evaluated when the eipd, ptfd, and pupsd daemons are started. If configuration changes are made during operation, the daemons have to be forced as follows to re-evaluate the _upstab file:

- eipd daemon with /etc/rc2.d/S98eip stop; /etc/rc2.d/S98eip start
- ptfd daemon with /etc/rc2.d/S98ptf stop; /etc/rc2.d/S98ptf start
- pupsd daemon with /etc/rc2.d/S98pups stop; /etc/rc2.d/S98pups start

Comment lines in the _upstab file begin with the character #. Blanks or tabs may be placed in front of it, if necessary. Comment lines are also ignored.

SEE ALSO

ups(1M), eipd(8), ptfd(8), pupsd(8), pwr(8).

System Administrator's Reference Manual

acct(4)

NAME

acct - per-process accounting file format

SYNOPSIS

```
#include <sys/types.h>
#include <sys/acct.h>
```

DESCRIPTION

Files produced as a result of calling acct(2) have records in the form defined by <sys/acct.h>, whose contents are:

```
typedef ushort comp_t;          /* "floating point" */
                                /* 13-bit fraction, 3-bit exponent  */
struct acct
{
    char   ac_flag;             /* Accounting flag */
    char   ac_stat;             /* Exit status */
    uid_t  ac_uid;              /* Accounting user ID */
    gid_t  ac_gid;              /* Accounting group ID */
    dev_t  ac_tty;              /* control typewriter */
    time_t ac_btime;            /* Beginning time */
    comp_t ac_utime;            /* acctng user time in clock ticks */
    comp_t ac_stime;            /* acctng system time in clock ticks */
    comp_t ac_etime;            /* acctng elapsed time in clock ticks */
    comp_t ac_mem;              /* memory usage in clicks */
    comp_t ac_io;               /* chars trnsfrd by read/write */
    comp_t ac_rw;               /* number of block reads/writes */
    char   ac_comm[8];          /* command name */
};
extern struct acct  acctbuf;
extern struct vnode *acctp;     /* vnode of accounting file */
#define AFORK   01              /* has executed fork, but no exec */
#define ASU     02              /* used superuser privileges */
#define ACCTF   0300            /* record type: 00 = acct */
#define AEXPND  040             /* Expanded Record Type*/
```

In ac_flag, the AFORK flag is turned on by each fork and turned off by an exec. The ac_comm field is inherited from the parent process and is reset by any exec. Each time the system charges the process with a clock tick, it also adds to ac_mem the current process size, computed as follows:

(data size) + (text size) / (number of in-core processes using text)

The value of ac_mem/(ac_stime+ac_utime) can be viewed as an approximation to the mean process size, as modified by text sharing.

NOTES

The ac_mem value for a short-lived command gives little information about the actual size of the command, because ac_mem may be incremented while a different command (e.g. the shell) is being executed by the process.

SEE ALSO

acctcom(1), acct(1M), acct(2), exec(2), fork(2).

System Administrator's Reference Manual

admin(4)

NAME

admin - installation defaults file

DESCRIPTION

admin is a generic name for an ASCII file that defines default installation actions by assigning values to installation parameters. For example, it allows administrators to define how to proceed when the package being installed already exits on the system.

/var/sadm/install/admin/default is the default admin file delivered with Reliant UNIX. The default file is not writable, so to assign values different from this file, create a new admin file. There are no naming restrictions for admin files. Name the file when installing a package with the –a option of pkgadd. If the –a option is not used, the default admin file is used.

Each entry in the admin file is a line that establishes the value of a parameter in the following form:

param=value

12 parameters can be defined in an admin file. A file is not required to assign values to all 12 parameters. If a value is not assigned, pkgadd asks the installer how to proceed.

The 12 parameters and their possible values are shown below except as noted. They may be specified in any order. Any of these parameters can be assigned the value ask, which means that if the situation occurs the installer is notified and asked to supply instructions at that time.

basedir
: Defines the base directory in which collectively relocatable objects should be installed. The following specifications for *value* are supported:

  dir The base directory is specified directly. The value can contain the parameter $PKGINST to specify a base directory which is independent of the program package concerned. *dir* must be an absolute pathname, i.e. it must begin with "/".

  default
  : The base directory for collectively relocatable objects is retrieved (if present) from the pkginfo(4) file of the package to be installed.

  ask (or empty)
  : The base directory for collectively relocatable objects is requested interactively (if interaction is permitted).

  Prerequisites for basedir becoming effective are generally

  - the existence of freely relocatable objects
  - the presetting of $BASEDIR variables in the pkginfo(4) file with a non-zero standard value

mail Defines a list of users to whom mail should be sent following installation of a package. If the list is empty, no mail is sent. If the parameter is not present in the admin file, the default value of root is used. The ask value cannot be used with this parameter.

runlevel
: Indicates resolution if the run level is not correct for the installation or removal of a package. The following specifications for *value* are supported:

  nocheck
  : Does not check for run level.

  quit Aborts installation if run level is not reached.

conflict
: Specifies what to do if an installation expects to overwrite a previously installed file, thus creating a conflict between packages. The following specifications for *value* are supported:

`nocheck`
Does not check for conflict; files in conflict will be overwritten.

`quit` Aborts installation if conflict is detected.

`nochange`
Overrides installation of conflicting files; they will not be installed.

`setuid`
Checks for executables which will have setuid or setgid bits enabled after installation. The following specifications for *value* are supported:

`nocheck`
Does not check for setuid executables.

`quit` Aborts installation if setuid processes are detected.

`nochange`
Overrides installation of setuid processes; processes will be installed without setuid bits enabled.

`action`
Determines if action scripts provided by package developers contain possible security impact. The following specifications for *value* are supported:

`nocheck`
Ignores security impact of action scripts.

`quit` Aborts installation if action scripts may have a negative security impact.

`partial`
Checks to see if a version of the package is already partially installed on the system. The following specifications for *value* are supported:

`nocheck`
Does not check for a partially installed package.

`quit` Aborts installation if a partially installed package exists.

`instance`
Determines how to handle installation if a previous version of the package (including a partially installed instance) already exists. The following specifications for *value* are supported:

`quit` Exits without installing if an instance of the package already exists (does not overwrite existing packages).

`overwrite`
Overwrites an existing package if only one instance exists. If there is more than one instance, but only one has the same architecture, it overwrites that instance. Otherwise, the installer is prompted with existing instances and asked which to overwrite.

`unique`
Does not overwrite an existing instance of a package. Instead, a new instance of the package is created. The new instance will be assigned the next available instance identifier.

`idepend`
Controls resolution if other packages depend on the one to be installed. The following specifications for *value* are supported:

`nocheck`
Does not check package dependencies.

`quit` Aborts installation if package dependencies are not met.

`rdepend`
Controls resolution if other packages depend on the one to be removed. The following specifications for *value* are supported:

`nocheck`

Does not check package dependencies.

`quit` Aborts removal if package dependencies are not met.

`space`
Controls resolution if disk space requirements for package are not met. The following specifications for *value* are supported:

`nocheck`
Does not check space requirements (installation fails if it runs out of space).

`quit` Aborts installation if space requirements are not met.

`list_files`
`nocheck`Processed files are not listed. Otherwise yes. \{\

`list_files`
□

`nocheck`
Processed files are not listed. Otherwise yes.

NOTES

The value `ask` should not be defined in an `admin` file that will be used for non-interactive installation (since by definition, there is no installer interaction). Doing so causes installation to fail when input is needed.

EXAMPLE

```
basedir=default
runlevel=quit
conflict=quit
setuid=quit
action=quit
partial=quit
instance=unique
idepend=quit
rdepend=quit
space=quit
list_files=nocheck
```

SEE ALSO

`pkginfo(4)`.

System Administrator's Reference Manual

aliases(4)

NAME

aliases - aliases file for sendmail

SYNOPSIS

/etc/aliases

DESCRIPTION

This file describes user ID aliases used by /usr/sbin/sendmail. The file resides in /etc and is formatted as a series of lines of the form

```
name: name_1, name2, name_3, ...
```

The name is the name to alias, and the name_n are the aliases for that name. Lines beginning with white space are continuation lines. Lines beginning with # are comments.

Aliasing occurs only on local names. Loops cannot occur, since no message will be sent to any person more than once.

After aliasing has been done, local and valid recipients who have a .forward file in their home directory have messages forwarded to the list of users defined in that file.

This is only the raw data file; the actual aliasing information is placed into a binary format in the file /etc/aliases.db using the program newaliases(1M). A newaliases command should be executed each time the aliases file is changed for the change to take effect.

NOTES

If you have compiled sendmail with DBM support instead of NEWDB, you may have encountered problems in dbm(3X) restricting a single alias to about 1000 bytes of information. You can get longer aliases by "chaining"; that is, make the last name in the alias be a dummy name which is a continuation alias.

SEE ALSO

newaliases(1M), sendmail(1M), dbm(3X).

System Administrator's Reference Manual

archives(4)

NAME

archives - device header file

SYNOPSIS

usr/include/archives.h

DESCRIPTION

```
/* Magic numbers */
#define CMN_ASC 0x070701  /* Cpio Magic Number for -c header */
#define CMN_BIN 070707    /* Cpio Magic Number for Binary header */
#define CMN_BBS 0143561   /* Cpio Magic Number for Byte-Swap header */
#define CMN_CRC 0x070702  /* Cpio Magic Number for CRC header */
#define CMS_ASC "070701"  /* Cpio Magic String for -c header */
#define CMS_CHR "070707"  /* Cpio Magic String for odc header */
#define CMS_CRC "070702"  /* Cpio Magic String for CRC header */
#define CMS_LEN 6         /* Cpio Magic String length */
/* Various header and field lengths */
#define CHRSZ   76    /* -H odc size minus filename field */
#define ASCSZ   110   /* -c and CRC hdr size minus filename field */
#define TARSZ   512   /* TAR hdr size */
#define HNAMLEN 256   /* maximum filename length for binary and odc headers */
#define EXPNLEN 1024  /* maximum filename length for -c and CRC headers */
#define HTIMLEN 2     /* length of modification time field */
#define HSIZLEN 2     /* length of file size field */
/* cpio binary header definition */
struct hdr_cpio
{
    short  h_magic,              /* magic number field */
           h_dev;                /* file system of file */
    ushort h_ino,                /* inode of file */
           h_mode,               /* modes of file */
           h_uid,                /* uid of file */
           h_gid;                /* gid of file */
    short  h_nlink,              /* number of links to file */
           h_rdev,               /* maj/min numbers for special files */
           h_mtime[HTIMLEN],     /* modification time of file */
           h_namesize,           /* length of filename */
           h_filesize[HSIZLEN];  /* size of file */
    char   h_name[HNAMLEN];      /* filename */
};
/* cpio -H odc header format */
struct c_hdr
{
    char c_magic[CMS_LEN],
         c_dev[6],
         c_ino[6],
         c_mode[6],
         c_uid[6],
         c_gid[6],
         c_nlink[6],
         c_rdev[6],
         c_mtime[11],
         c_namesz[6],
         c_filesz[11],
```

```
        c_name[HNAMLEN];
};
/* -c and CRC header format */
struct Exp_cpio_hdr
{
    char E_magic[CMS_LEN],
         E_ino[8],
         E_mode[8],
         E_uid[8],
         E_gid[8],
         E_nlink[8],
         E_mtime[8],
         E_filesize[8],
         E_maj[8],
         E_min[8],
         E_rmaj[8],
         E_rmin[8],
         E_namesize[8],
         E_chksum[8],
         E_name[EXPNLEN];
};
/* Tar header structure and format */
#define TBLOCK  512  /* length of tar header and data blocks */
#define TNAMLEN 100  /* maximum length for tar file names */
#define TMODLEN 8    /* length of mode field */
#define TUIDLEN 8    /* length of uid field */
#define TGIDLEN 8    /* length of gid field */
#define TSIZLEN 12   /* length of size field */
#define TTIMLEN 12   /* length of modification time field */
#define TCRCLEN 8    /* length of header checksum field */
/* tar header definition */
union tblock
{
    char dummy[TBLOCK];
    struct header
    {
        char t_name[TNAMLEN];      /* name of file */
        char t_mode[TMODLEN];      /* mode of file */
        char t_uid[TUIDLEN];       /* uid of file */
        char t_gid[TGIDLEN];       /* gid of file */
        char t_size[TSIZLEN];      /* size of file in bytes */
        char t_mtime[TTIMLEN];     /* modification time of file */
        char t_chksum[TCRCLEN];    /* checksum of header */
        char t_typeflag;           /* flag to indicate type of file */
        char t_linkname[TNAMLEN];  /* file this file is linked with */
        char t_magic[6];           /* magic string always "ustar" */
        char t_version[2];         /* version strings always "00" */
        char t_uname[32];          /* owner of file in ASCII */
        char t_gname[32];          /* group of file in ASCII */
        char t_devmajor[8];        /* major number for special files */
        char t_devminor[8];        /* minor number for special files */
        char t_prefix[155];        /* pathname prefix */
    } tbuf;
};
/* volcopy tape label format and structure */
#define VMAGLEN 8
#define VVOLLEN 6
#define VFILLEN 464
struct volcopy_label
```

```
{
    char v_magic[VMAGLEN],
         v_volume[VVOLLEN],
         v_reels,
         v_reel;
    long v_time,
         v_length,
         v_dens,
         v_reelblks,       /* u370 added field */
         v_blksize,        /* u370 added field */
         v_nblocks;        /* u370 added field */
    char v_fill[VFILLEN];
    long v_offset;         /* used with -e and -reel options */
    int  v_type;           /* does tape have nblocks field? */
};
```

System Administrator's Reference Manual

binarsys(4)

NAME

`binarsys -` remote system information for the ckbinarsyscommand

SYNOPSIS

`/etc/mail/binarsys`

DESCRIPTION

`binarsys` contains lines of the form:

remote_system_name`:`*val*

where *val* is either `Y` or `N`. This line indicates whether that particular remote system can deal properly with messages that have a binary content. The absence of an entry for a particular system or absence of the `binarsys` file altogether will imply No.

Blank lines or lines beginning with `#` are considered comments and ignored. Should a line of `Default=y` be encountered, the default condition for missing entries described in the previous paragraph is reversed to be `Y`es. Another line of `Default=`*n* will restore the default condition to No.

`mail` is distributed with the `binarsys` file containing only a `Default=y` line.

SEE ALSO

`mail`(1), `ckbinarsys`(1M), `mailsurr`(4).

System Administrator's Reference Manual

checkpasswd(4)

NAME

`checkpasswd` - /etc/default/checkpasswd file

DESCRIPTION

The `/etc/default/checkpasswd` file contains additional settings for the `passwd(1)` command. These settings also apply for `root`.

This file is only evaluated by the `passwd(1)` command if the `ADDCHECK=yes` variable is set in the `/etc/default/passwd` file.

FORMAT

This format of this file is *identifier*<TAB>*value*. A blank can also be used as a separator instead of the tab character.

CONTENTS

`dictionary`<TAB>*path_to_dictionary*
: Pathname of the dictionary containing the illegal passwords [refer to `makedict(1M)`]. This entry can occur several times.

`singlecase`<TAB>`yes` | `no`
: Only lowercase or only uppercase is permitted.

`printonly`<TAB>`yes` | `no`
: Only printable characters are permitted.

`badchars`<TAB>`"`*string*`"`
: Replaces the list of illegal non-printable characters [cf. `passwd(1)`] with the characters in *string*.

`badchars`<TAB>`+"`*string*`"`
: Adds the characters in *string* to the list of illegal non-printable characters [cf. `passwd(1)`].

DEFAULT VALUES

There are no defaults in this file.

NOTES

The non-printable characters in the `badchars` entry must be entered in the form `^`*X*, thus for example `^L` for `CTRL-L` [=□FF character (Form Feed), see `ascii(5)`].

SEE ALSO

`passwd(1)`, `makedict(1M)`, `default_passwd(4)`.

System Administrator's Reference Manual

choices(4)

NAME

choices - terminal setting file

SYNOPSIS

/var/sadm/terms/choices

DESCRIPTION

The choices file contains tables describing all the types of terminal that can be selected, with their country variants and keyboard variants. Each entry listed in the file represents a valid combination of terminal type, country variant and keyboard variant.

The file is used by the commands termadd(1M) and termshow(1M) to determine from the parameters specified by the user, the terminal software, terminal variable and the mark for the line parameters. The file is structured as follows:

TYPE:COUNTRY:KEYBOARD:TERM:SOFTWARE:LABEL:TAG:

The entries are separated with ":". Lines that begin with "#" are interpreted as comment lines.

The entry TYPE determines the terminal type. COUNTRY indicates the country code, KEYBOARD the keyboard variant. If "default" is used as the values for the BA80 for COUNTRY or KEYBOARD and SOFTWARE, the terminal determines the country or keyboard variants to be loaded.

TERM determines the terminal variable to be used ($TERM). SOFTWARE is the name of the software required to operate the terminal (e.g. BA80 workstation program or code table). In general, the SOFTWARE is directly dependent on COUNTRY and KEYBOARD.

LABEL is the mark for the line parameters in /etc/ttydefs. TAG indicates the device:

- T Terminal
- P Printer
- S Serial
- M MWSS workstation

termadd determines from the file the suitable entry for the required terminal. If the keyboard and country variant are not specified, the first suitable entry (with any keyboard or country variant) is taken from the file.

NOTES

An inconsistent file may inhibit the use of the termadd or termshow commands. Thus, the commands should not be executed out by the user.

The description of a combination must not exceed one line.

EXAMPLE

```
# ba47 Terminals; CT03
#
ba47:default:default:ba47:default:ba80:T:
ba47:default:ct03:ba47:default:ba80:T:
```

SEE ALSO

termadd(1M), termdel(1M), termshow(1M).

System Administrator's Reference Manual

compver(4)

NAME

`compver` - compatible versions file

DESCRIPTION

`compver` is an ASCII file used to specify previous versions of the associated package which are upwards compatible. It is created by a package developer.

Each line of the file specifies a previous version of the associated package with which the current version is backwards compatible.

Since some packages may require installation of a specific version of another software package, compatibility information is extremely crucial. Consider, for example, a package called A which requires version "1.0" of package B as a prerequisite for installation. If the customer who installed package A now has a newer version of B (Version 1.3), the `compver` file for B must contain an entry "1.0", which indicates that Version "1.3" from the `pkginfo(4)` file of B is compatible with Version "1.0" of B, so that the customer can install package A.

NOTES

The comparison of the version string disregards white space and tabs. It is performed on a word-by-word basis. Thus "4 1.3" and "4□1.3" would be considered the same.

EXAMPLE

A sample `compver` file is shown below.

```
1.2
1.0
```

If this package contains the entry `VERSION="1.3"` in `pkginfo(4)`, versions "1.3", "1.2" and "1.0" are compatible.

SEE ALSO

`depend(4)`, `pkginfo(4)`.

System Administrator's Reference Manual

controller(4)
RM600 only

NAME

`controller` - controller configuration file

SYNOPSIS

`/etc/controller`

DESCRIPTION

The `/etc/controller` file is an ASCII file in which logical numbers are assigned to all I/O controllers of the system.

If an external MBII has a dual MBP, then each controller on this MBII bus can also be assigned an MBP of its MBII bus in `/etc/controller`, in order to distribute the load. The assignment is made by entering the logical number of the MBP required in the field `mbp` (see below).

The configuration defined in `/etc/controller` first takes effect in the system when `ldcontroller(8)` is called and the system has been rebooted. `ldcontroller` is called each time the system is brought down. Modifications to `/etc/controller` should not be made directly, rather by calling `modcontroller(8)` so that the consistency of `/etc/controller` is guaranteed.

If the `/etc/controller` file does not exist when the system is brought down, it is automatically created by `mkcontroller(8)` before `ldcontroller(8)` is called.

If existing controllers are not entered in `/etc/controller`, they are automatically assigned a logical number when the system is started. They are therefore available for the device drivers. If there is no `mbp` specification, the load for the controllers is distributed according to a standard algorithm.

If a controller is entered in `/etc/controller` that does not physically exist when the system is started, the logical number assigned to it is not assigned to any other controller, i.e. a controller retains its logical number even if it has been temporarily removed, for example for repairs.

Each I/O controller of the system bus has a line in the file `/etc/controller`, where the individual fields are separated by the ":" character. The line is structured as follows:

`ctlr_type : log_no : bus_type : cab_id : bus_id : slot_id [: chan [: mbp \`
`[: option]]]`

Where the fields have the following meaning:

`ctrl_type`
: Controller type (e.g. `lce`, `scsi`; according to the definition of the device drivers)

`log_no`
: Logical number of the controller as a decimal number (0 ... 99; unique within a controller type)

`bus_type`
: Type of the relevant bus (SP, MBII)

`cab_id`
: Number of the cabinet containing the controller (0 ... 15; system cabinet 0)

`bus_id`
: RM600 with R10.000 CPU: bus number assigned by the system
RM600-xxx: empty

`slot_id`
: Number of the slot on the bus (0 ... 17)

`chan` Channel number of the controller (0 ... 6 or empty)

`mbp` Logical number of the MBP to be used by this controller to receive jobs (0 ... 14; must be on the same MBII)

`option`

Driver specific optional number (0 ... 254) (e.g. SCSI-ID of DCS controller)

Lines with the "#" character in the first column are interpreted as comment and are ignored by `ldcontroller(8)`.

SEE ALSO

`ldcontroller(8)`, `mkcontroller(8)`, `modcontroller(8)`, `showconf(8)`.

System Administrator's Reference Manual

copyright(4)

NAME

`copyright` - copyright information file

DESCRIPTION

`copyright` is an ASCII file used to provide a copyright notice for a package. The text may be in any format. The full file contents (including comment lines) is displayed on the terminal at the time of package installation.

If there is a copyright file for a package, it must be identified as an information file in the package's `prototype` file.

SEE ALSO

`prototype`(4) .

System Administrator's Reference Manual

cron(4)

NAME

`cron - /etc/default/cron` file

DESCRIPTION

The `/etc/default/cron` file contains the default settings for the `cron(1M)` command.

FORMAT

This format of this file is *identifier=value* where both the identifiers and values are defined by the `cron(1M)` program.

CONTENTS

`CRONLOG`
: If set to `YES` then all commands run by `cron(1M)` are logged in `/var/cron/log`.

`MAXRUN`
: The optional configuration is a value in the range of 10 to 5120, for specifying the maximum number of jobs that can be run simultaneously spawned by `cron(1M)` .

DEFAULT VALUES

```
CRONLOG=YES
MAXRUN=25
```

SEE ALSO

`cron(1M)` .

default_passwd(4) - fspec(4)

System Administrator's Reference Manual

default_passwd(4)

NAME

`default_passwd - /etc/default/passwd` file

DESCRIPTION

The `/etc/default/passwd` file contains the default settings for the `passwd(1)` command.

FORMAT

This format of this file is *identifier=value*.

CONTENTS

`HISTORY`
: Number of passwords last used, which are locked for the `passwd(1)` password. The passwords must not be the same.

`MAXWEEKS`
: The maximum number of weeks for which the password is valid.

`MINWEEKS`
: The minimum number of weeks required between password changes.

`WARNWEEKS`
: The number of weeks relative to `MAXWEEKS` before the password expires that the user will be warned.

`PASSLENGTH`

The minimum number of characters in a password.

ADDCHECK
: If ADDCHECK=yes, additional checks are carried out for which the /etc/default/checkpasswd file is evaluated [refer to checkpasswd(4)].

DOMAINPASS
: If DOMAINPASS=yes, the passwd(1) command on the same machine can set a global password for a user (i.□e. a password that is valid on all machines in domains managed by SIclos [see clos(7)] or WSA [see wsa(1M)]). If DOMAINPASS is not set or contains the value no, the domain-wide functionality in passwd is disabled.

Warning:

If the SIclos package is not installed and DOMAINPASS is set to yes, use of the –g option in the passwd(1) command has no effect.

DEFAULT VALUES

```
HISTORY=0
MAXWEEKS=24
MINWEEKS=0
WARNWEEKS=1
PASSLENGTH=6
ADDCHECK=no
DOMAINPASS=no
```

SEE ALSO

passwd(1) , checkpasswd(4) .

System Administrator's Reference Manual

depend(4)

NAME

`depend` - software dependencies files

DESCRIPTION

`depend` is an ASCII file used to specify information concerning software dependencies for a particular package. The file is created by a software developer.

Each entry in the `depend` file describes a dependency with one software package. The format of each entry is:

type pkg name
[(*arch*)]*version*
[(*arch*)]*version*
...

The fields are:

type Defines the dependency type. Must be one of the following characters:

- `P` Indicates a prerequisite for installation, for example, the referenced package or versions must be installed.
- `I` Implies that the existence of the indicated package or version is incompatible.
- `R` Indicates a reverse dependency. Instead of defining the package's own dependencies, this designates that another package depends on this one. This type should be used only when an old package does not have a `depend` file but it relies on the newer package nonetheless. Therefore, the present package should not be removed if the designated old package is still on the system since, if it is removed, the old package will no longer work.

pkg Indicates the package abbreviation.

name Specifies the full package name.

[(*arch*)]*version* or ~*version*
Specifies a particular package instance to which the dependency applies. The instance is identified by the package version and where necessary by the package architecture (in parentheses). Each instance specification must begin on a new line that begins with a blank or tab. A null version set equates to any version of the indicated package.

If the version specification in the `depend` file of a dependent package is preceded by a tilde (`~`), the `compver(4)` file is examined as well. That means that the `compver` file must be maintained in the package whose version is changing, and the new version must be compatible with the earlier version.

EXAMPLE

Here is an example of the relationship between the `depend` file and the `compver` file:

Package A, `compver` file:

```
1.0
2.0
3.0
```

This means that `VERSION` in `pkginfo(4)` may be followed by 1.0, 2.0 or 3.0. These versions are compatible.

Package B, `depend` file:

```
P lp Line Printer Spooling Utilities
  ~1.0
```

In this case package A's `compver` file will be examined when package B's dependencies are evaluated.

SEE ALSO

compver(4), pkginfo(4).

System Administrator's Reference Manual

dir(4-ufs)

NAME

`dir(ufs)` - format of ufs directories

SYNOPSIS

```
#include <sys/param.h>
#include <sys/types.h>
#include <sys/fs/ufs_fsdir.h>
```

DESCRIPTION

A directory consists of some number of blocks of `DIRBLKSIZ` bytes, where `DIRBLKSIZ` is chosen such that it can be transferred to disk in a single atomic operation (e.g. 512 bytes on most machines).

Each `DIRBLKSIZ`-byte block contains some number of directory entry structures, which are of variable length. Each directory entry has a `struct direct` at the front of it, containing its inode number, the length of the entry, and the length of the name contained in the entry. These are followed by the name padded to a 4-byte boundary with null bytes. All names are guaranteed null-terminated. The maximum length of a name in a directory is `MAXNAMLEN`.

```
#define DIRBLKSIZ DEV_BSIZE
#define MAXNAMLEN 256
struct direct
{
    u_long  d_ino;                  /* inode number of entry */
    u_short d_reclen;               /* length of this record */
    u_short d_namlen;               /* length of string in d_name */
    char    d_name[MAXNAMLEN + 1];  /* name must be no longer than this */
};
```

SEE ALSO

`fs`(4-ufs) .

System Administrator's Reference Manual

direct(4)
BSD Compatibility Package

NAME

`direct` - file system independent directory entry

SYNOPSIS

```
#include <direct.h>
```

DESCRIPTION

Different file system types may have different directory entries. The `direct` structure defines a file system independent directory entry, which contains information common to directory entries in different file system types. A set of these structures is returned by the `readdir`(3C) system call.

The `direct` structure is defined in this example:

```
struct  direct {
        u_long  d_ino;
        u_short d_reclen;
        u_short d_namlen;
        char    d_name[NAME_MAX + 1];
};
```

The `d_ino` is a number which is unique for each file in the file system. The field `d_namelen` is the length of the string in `d_name`. The field `d_name` is the beginning of the character array giving the name of the directory entry. This name is null terminated and may have at most `NAME_MAX` characters. This results in file system independent directory entries being variable length entities. The value of `d_reclen` is the record length of this entry. This length is defined to be the number of bytes between the current entry and the next one, so that the next structure will be suitably aligned.

SEE ALSO

`readdir`(3C) .

System Administrator's Reference Manual

dirent(4)

NAME

`dirent` - file system independent directory entry

SYNOPSIS

```
#include <dirent.h>
```

DESCRIPTION

Different file system types may have different directory entries. The `dirent` structure defines a file system independent directory entry, which contains information common to directory entries in different file system types. A set of these structures is returned by the `getdents(2)` system call.

The `dirent` structure is defined in this example:

```
struct dirent
{
    ino_t           d_ino;          /* "inode number" of entry */
    off_t           d_off;          /* offset of disk directory entry */
    unsigned short  d_reclen;       /* length of this record */
    char            d_name[1];      /* name of file */
};
```

The type `ino_t` is defined as described in `<sys/types.h>`.

The `d_ino` is a number which is unique for each file in the file system. The field `d_off` is the offset of that directory entry in the actual file system directory. The field `d_name` is the beginning of the character array giving the name of the directory entry. This name is null terminated and may have at most `NAME_MAX` characters. This results in file system independent directory entries being variable length entities. The value of `d_reclen` is the record length of this entry. This length is defined to be the number of bytes between the current entry and the next one, so that the next structure will be suitably aligned.

The following are declared as functions and may also be defined as macros:

`int` `closedir(DIR *`*dirp*`);`

`DIR` `*opendir(const char *`*dirname*`);`

`struct dirent` `*readdir(DIR *`*dirp*`);`

`int` `readdir_r(DIR *`*dirp*, struct direct **entry*, struct dirent ***result*);

`void` `rewinddir(DIR *`*dirp*`);`

`void` `seekdir(DIR *`*dirp*, long int *loc*);

`long int` `telldir(DIR *`*dirp*`);`

The `dirent64` structure is defined in the same way as the `dirent` structure with the exception of the following members:

```
    ino64_t         d_ino;          /* "inode number" of entry */
    off64_t         d_off;          /* offset of disk directory entry */
```

The following is declared as a function and may also be defined as a macro:

`struct dirent64 *readdir64(DIR *`*dirp*`);`

SEE ALSO

`getdents(2)`, `directory(3C)`, `lfs(5)`, `types(5)`.

System Administrator's Reference Manual

dktab(4)

NAME

`dktab` - virtual disk configuration file

SYNOPSIS

`/etc/dktab`

DESCRIPTION

The `/etc/dktab` file continues to exist alongside the new Vdisk Lite (VL) procedure, where the configuration information for the vdisks (virtual disks) is stored in special system databases. This file describes the configurations of the vdisks which were not created using `vdadmin(8)`. The `dkconfig(8)` command first evaluates the VL vdisks and then those described in `/etc/dktab`. For example, if one first creates a vdisk using `vdadmin` and then later enters a description of the same vdisk in `/etc/dktab`, the VL version has precedence. However, `dkconfig` issues an appropriate warning.

A detailed description of virtual disks as well as of Vdisk Lite (VL) can be found under `vdisk(7)` or `vl(7)`.

The system administrator is responsible for maintaining this file. This file should be well-protected (similar to the protection given to the `/etc/vfstab` file), with only `root` write permission allowed.

The system administrator can modify this file with a text editor. Within this file, fields are separated by white space. "#" as the first character of a line indicates a comment. Blank lines are ignored. The order of virtual disks in `/etc/dktab` is not significant unless nested virtual disks are defined.

All pathnames used in `/etc/dktab` are block special devices. The corresponding character special devices follow the configuration of the block special devices.

All offsets and sizes are specified in virtual disk block units. Each virtual disk block is 512 bytes long. The sectors of the underlying physical disk may be one or more virtual disk block units in size. The offsets and sizes must be an integral number of physical disk sectors.

VIRTUAL DISK CONFIGURATION

Virtual disks require two types of configuration lines in the `/etc/dktab` file: virtual disk declaration lines and virtual piece definition lines. The file consists of sets of a single virtual disk declaration line followed by N virtual piece definition lines. Virtual disk declaration lines begin with the first character of a line, whereas virtual piece definition lines are preceded with a white space such as a tab or blank.

The following is an example of an `/etc/dktab` file segment for virtual disks:

```
# A concatenated virtual partition.
/dev/vd/vdisk0 concat 4
    /dev/ios0/sdisk001s4
    /dev/ios0/sdisk001s5
    /dev/ios0/sdisk002s4
    /dev/ios0/sdisk002s5
# A striped virtual partition with a cluster size of 32 blocks.
/dev/vd/vdisk1 stripe 2 32
    /dev/ios0/sdisk001s4
    /dev/ios0/sdisk002s4
# Three tiny 2400 block simple virtual partitions.
/dev/vd/vdisk2 simple
    /dev/ios0/sdisk010s7    0       2400
/dev/vd/vdisk3 simple
    /dev/ios0/sdisk010s7    2400    2400
/dev/vd/vdisk4 simple
    /dev/ios0/sdisk010s7    4800    2400
# Preallocated memory virtual disk.
/dev/vd/vdisk9 memory    2000
```

```
# Lazy allocation (allocation by demand) memory virtual disk.
/dev/vd/vdisk10   vmemory   2000
```

Virtual Disk Declaration Lines

A virtual disk declaration line always has at least two fields: The pathname of the virtual disk and its type. Valid virtual disk types for the `vdisk` product are "simple", "concat", "stripe", and "memory" or "vmemory". The `mirror` product additionally supports the types "mirror", "statesave", and "statsv" [see `mirror(7)`].

Concatenated, striped, and memory virtual disks require a third field. For concatenated and striped virtual disks, this is the number of pieces from which the virtual disk is composed. For memory virtual disks, this is the size to be allocated (in blocks). Striped virtual disks require a fourth field, which is the cluster size in 512-byte blocks.

The following are examples of virtual disk declaration lines:

```
/dev/vd/vdisk0  simple
/dev/vd/vdisk1  concat  2
/dev/vd/vdisk2  stripe  2       32
/dev/vd/vdisk4  memory    2000
/dev/vd/vdisk5  vmemory   32768
```

Virtual Piece Definition Lines

A virtual piece definition line describes one piece of a virtual disk and consists of one or three fields. The first field is required and is most often the pathname of a block special physical disk device. The other two optional fields are the block offset and length of the physical disk device in 512-byte blocks. If given, only the specified length beginning at the specified offset will be used as the piece of the virtual disk; otherwise, the entire physical disk partition is assumed. No piece definition is needed for the memory virtual disk.

A virtual piece definition line may also consist of the pathname of a block special virtual disk device, if a nested virtual disk definition is desired. Note, however, that this virtual disk device must exist as a previously-defined and configured entry in the `/etc/dktab` file.

Here are examples of virtual piece definition lines:

```
/dev/ios0/sdisk001s1
/dev/ios0/sdisk001s2    0       12000
/dev/ios0/sdisk001s3    25200   50000
/dev/vd/vdisk14
```

CONFIGURATION OF THE MIRROR DISKS OF AN OLR GROUP

OLR groups can only be defined for SCSI channels not for fiber channels.

This section assumes a knowledge of mirror disks as well as statesave disks. See also `vdisk(7)` and `mirror(7)` .

1) The SCSI tiers used do not support the "hot swap" feature

An alternative form of mirror disk configuration was defined for the OLR (online replacement) functionality. One OLR group is spread out over two or three SCSI ports. These ports must be identically structured in terms of the type of disks connected and their SCSI addresses. Partitions are mirrored in the usual way, however in this concept only the same partitions of disks with the same SCSI address can be mirrored on different SCSI ports, e.g. `/dev/ios0/sdisk032s4` with `/dev/ios0/sdisk052s4`. It is also not possible to specify offset or length. The disks to be mirrored therefore must have the same partition table.

It is important that all disks are mirrored to the relevant ports. Access to the physical partitions of the disks is no longer possible following the configuration of the first OLR-mirror disk (error `EBUSY`).

The definition of an OLR group begins with the keyword `olr`, followed by the declaration of the group number and the number N of pieces of the subsequently defined mirror disks. N can be 2 or 3. The next N lines determine the SCSI ports to which the disks are connected. The following lines each contain the definition of a virtual disk. The first virtual disk must be of the type "statesave", the next of the type "mirror". Such a definition

line has the following structure:

Name of the virtual disk, its type, disk and partition.

An example:

```
olr,group=1 2
        ios0/sport03
        ios0/sport06
        vdisk10 statesave disk0s15
        vdisk0  mirror  disk0s7
        vdisk2  mirror  disk1s6
        vdisk3  mirror  disk1s8
```

In "old" notation this corresponds to:

```
/dev/vd/vdisk10 statesave,group=1 2
 /dev/ios0/sdisk030s15
 /dev/ios0/sdisk060s15
/dev/vd/vdisk0 mirror,group=1 2
 /dev/ios0/sdisk030s7
 /dev/ios0/sdisk060s7
/dev/vd/vdisk2 mirror,group=1 2
 /dev/ios0/sdisk031s6
 /dev/ios0/sdisk061s6
/dev/vd/vdisk3 mirror,group=1 2
 /dev/ios0/sdisk031s8
 /dev/ios0/sdisk061s8
```

2) The SCSI tiers used support the "hot swap" feature

If SCSI tiers are used that support a "hot-swap" (in the BG70/BG71 auxiliary cabinets), it is not necessary for the entire SCSI bus to be idle in order to exchange a disk. In this case, you simply need to ensure that all activities on the disk to be exchanged are stopped. The keyword `olr` was introduced for this reason for individual mirror disks. An example follows:

```
/dev/vd/vdisk10 statesave,olr,group=1 2
     /dev/ios0/sdisk030s15
     /dev/ios0/sdisk060s15
/dev/vd/vdisk0 mirror,olr,group=1     2
     /dev/ios0/sdisk030s6
     /dev/ios0/sdisk060s6
```

After this mirror disk has been configured, all partitions on the `ios0/sdisk030` and `ios0/sdisk060` physical disks cannot be accessed directly. Other physical disks on the `ios0/sport03` and `ios0/sport06` SCSI ports are not affected.

CONFIGURATION OF VIRTUAL DISKS IN A DLM (OPS) ENVIRONMENT

If the product OPS (= Oracle Parallel Server) is implemented, distributed mirror disks are used. Consistency of the status of this type of mirror disk is maintained on the participating systems (RM600/RM400).

The keyword `ops` is sufficient for configuration of a mirror disk. Example:

```
/dev/vd/vdisk0    statesave,ops,group=0  2
 /dev/ios0/sdisk002s15   0       32
 /dev/ios0/sdisk003s15   0       32
/dev/vd/vdisk1  mirror,ops,group=0      2
        /dev/ios0/sdisk002s2
        /dev/ios0/sdisk003s2
```

The keyword `ops` in the type field indicates that the virtual disk is a distributed mirror disk (type "mirror"). The keyword `ops` must also be contained in the corresponding definition of the virtual disk of type "statesave". The definition for the distributed disks (type "statesave" and "mirror") must be identical on all hosts in domain 9 of the DLM configuration (see the DLM manual). The keyword `opsres` is used instead of `ops` in order to activate the "Oracle resilvering" function for an OPS mirror disk.

It is also possible to combine the functions `ops` and `olr`. Example:

```
olr,ops,group=3 2
 ios0/sport02
 ios0/sport04
 vdisk100 statesave disk0s15
 vdisk101 mirror    disk0s7
 vdisk102 mirror    disk1s7
```

In this case, distributed mirror disks are defined whose physical disks are "online replaceable".

Virtual disks with the attribute `ops` are configured by the startup script `/etc/rc2.d/S85dlm` at system startup. All virtual disks (except of type "memory" and "vmemory") can have the `ops` attribute. However this causes these virtual disks to be configured automatically at system startup.

CONFIGURATION OF MIRROR DISKS IN AN OBSERVE ENVIRONMENT

If you find yourself in an Observe environment, that is, one where disk containers are taken over by a backup system following system failure, this can lead to problems because the SCSI paths to the disks change. This can happen, for example, if the master system is an RM600 and the backup system is an RM400, as the RM600 uses the path `ios0/sdisk`*xxx*, while the RM400 uses `ios1/sdisk`*xxx*.

If the mirror disks are taken over by the RM400, it can be established using the statesave information that the names of the physical devices have changed. In such a case, the statesave driver outputs the following message: "Configuration changed since last time, previous state no longer applicable", and switches all the partitions of the mirror disk to `disabled` status, although both were previously `online`. To bypass this behavior, the keyword `switch` is introduced for mirror disks. The effect of this is that only parts of the SCSI name of a physical device are checked, that is the SCSI ID and the partition number. Example:

RM600 master system:

```
/dev/vd/vdisk27 mirror,switch 2
 /dev/ios0/sdisk123s7
 /dev/ios0/sdisk246s7
```

RM400 backup system:

```
/dev/vd/vdisk27 mirror,switch 2
 /dev/ios1/sdisk073s7
 /dev/ios1/sdisk016s7
```

After the disks of the RM400 have been activated, the mirror disk is configured and its status is taken on by the RM600. The greatest advantage of this procedure is that there is no need to start a catchup process for all mirror disks after each deactivation.

If devices are configured which change their SCSI ID in the course of switching over (Raid devices, `rdswitch(8)` command), the keyword `nocheck` can be used. As the name implies, the entire device name is not checked if this option is used. Example:

RM600 master system:

```
/dev/vd/vdisk27 mirror,nocheck 2
 /dev/ios0/sraid086l0s5
 /dev/ios0/sraid096l0s5
```

RM600 backup system:

```
/dev/vd/vdisk27 mirror,nocheck 2
 /dev/ios0/sraid073l1s5
 /dev/ios0/sraid095l2s5
```

CONFIGURATION OF MIRROR DISKS IN AN RMS ENVIRONMENT

You can specify the `hv` option in the mirror disk definition for using mirror disks in an RMS (Reliant Monitoring Software) configuration. If this option is set and all pieces of this mirror disk have the status `disabled`, the mirror disk driver knows which piece last had this status.

You should invoke `dkmirror -R` to reset the status of this piece to `online`. This functionality is only

intended for the RMS status and is disabled with the OPS functionality.

Example:

```
/dev/vd/vdisk27 mirror,hv
 /dev/ios0/sraid073l1s5
 /dev/ios0/sraid095l2s5
```

VIRTUAL DISKS OF TYPE "mirror,readonly"

Virtual disks of type `mirror,readonly` are mirror disks which contain unchangeable data. What this means, in particular, is that the individual pieces of this mirror disk cannot contain different data. It is therefore not necessary with such mirror disks to perform a catchup because a piece of the disk is temporarily unavailable (e.g. because an FO segment has failed). In this instance, the piece that is temporarily out of service can be brought `online` again with

`dkmirror -C # /dev/vd/vdisk` ...

Syntax

The new `readonly` keyword can be entered in the `/etc/dktab` file. Possible syntax would be

```
/dev/vd/vdisk35 mirror,readonly 2
 /dev/ios0/sdisk032s6
 /dev/ios0/sdisk062s6
```

or when combined with other options:

```
/dev/vd/vdisk35 mirror,olr,readonly,group=7 2
 /dev/ios0/sdisk033s8
 /dev/ios0/sdisk063s8
```

The `readonly` keyword can also be assigned to other types of virtual disks ("simple", "concat" or "stripe"), though it has no meaning for other types ("statesave", "statsv", "memory"). `readonly` is also not supported in OLR groups.

FILES

`/etc/dktab`
: virtual disk configuration file

SEE ALSO

`mirror(7)`, `olr(7)`, `vdisk(7)`, `vl(7)`, `dkconfig(8)`.

System Administrator's Reference Manual

dumpsave(4)

NAME

dumpsave - /etc/default/dumpsave file

DESCRIPTION

The /etc/default/dumpsave file contains the default settings for the dumpsave(1M) command.

FORMAT

This format of this file is *identifier=value* where both the identifiers and values are defined by dumpsave(1M).

CONTENTS

DUMPDIR
: Specifies the system dump directory.

MINFREE
: Specifies the minimum number of 512-byte blocks that should be left free on the file system containing DUMPDIR. If the dump, or /unix, or both will not fit in DUMPDIR without leaving MINFREE blocks free, both the dump and /unix will not be saved.

UNIXFILE
: Typically UNIXFILE is /unix.

SWAPFILE
: Typically SWAPFILE is /dev/swap.

ULIMIT
: The default value of ULIMIT is unlimited; however, ULIMIT can be set lower.

TAPEDEV
: Specifies the absolute path name of a "no-rewind" tape device. If the memory dump does not fit in the directory specified with DUMPDIR, an attempt will be made to write the dump directly to this tape device.

  If TAPEDEV=none is set, the memory dump is not written to any tape device.

TIMELIMIT
: Specifies the time period (in seconds) that has to be waited for input from the user when accessing the tape device.

DEFAULT VALUES

```
DUMPDIR=/var/crash
MINFREE=4096
UNIXFILE=/unix
SWAPFILE=/dev/rswap
TAPEDEV=/dev/ios0/rstape004n
TIMELIMIT=300
```

SEE ALSO

dumpsave(1M).

System Administrator's Reference Manual

environ(4)

NAME

environ, pref, variables - user-preference variable "dot" files for FACE

DESCRIPTION

The .environ, .pref, and .variables files contain variables that indicate user preferences for a variety of operations. The .environ and .variables files are located under the user's $HOME/pref directory. The .pref files are found under $HOME/FILECABINET, $HOME/WASTEBASKET, and any directory where preferences were set via the organize command. Names and descriptions for each variable are presented below. Variables are listed one per line and are of the form *variable=value*.

Variables found in .environ include:

LOGINWIN[1-4]
: Windows that are opened when FACE is initialized

SORTMODE
: Sort mode for file folder listings. Values include the following hexadecimal digits:

  1 sorted alphabetically by name

  2 files most recently modified first

  800 sorted alphabetically by object type

  The values above may be listed in reverse order by "ORing" the following value:

  1000 list objects in reverse order. For example, a value of 1002 will produce a folder listing with files LEAST recently modified displayed first. A value of 1001 would produce a "reverse" alphabetical by name listing of the folder.

DISPLAYMODE
: Display mode for file folders. Values include the following hexadecimal digits:

  0 file names only

  4 file names and brief description

  8 file names, description, plus additional information

WASTEPROMPT
: Prompts before emptying wastebasket (yes/no)?

WASTEDAYS
: Number of days before emptying wastebasket.

PRINCMD[1-3]
: Prints command defined to print files.

UMASK
: Holds default permissions that files will be created with.

Variables found in .pref are the following:

SORTMODE
: which has the same values as the SORTMODE variable described in .environ above.

DISPMODE
: which has the same values as the DISPLAYMODE variable described in .environ above.

Variables found in .variables include:

EDITOR
: Default editor

PS1 Reliant UNIX shell prompt

FILES

```
$HOME/pref/.environ
$HOME/pref/.variables
$HOME/FILECABINET/.pref
$HOME/WASTEBASKET/.pref
```

System Administrator's Reference Manual

fd(4)

NAME

`fd` - /dev/fd file descriptor files

DESCRIPTION

These files, conventionally called `/dev/fd/0`, `/dev/fd/1`, `/dev/fd/2`, and so on, refer to files accessible through file descriptors. If file descriptor *n* is open, these two system calls have the same effect:

`fd = open("/dev/fd/`*n*",mode);
fd = dup(*n*);

For these files, `creat(2)` is equivalent to `open`, and `mode` is ignored. As with `dup`, subsequent reads or writes on `fd` fail unless the original file descriptor allows the operations.

For convenience in referring to standard input, standard output, and standard error, an additional set of names is provided: `/dev/fd/0` is a synonym for `stdin`, `/dev/fd/1` for `stdout`, and `/dev/fd/2` for `stderr`. In fact, `/dev/fd/`*n* is a synonym for any valid file descriptor *n* for that process.

NOTE

This is true only if `/dev/fd` is the mount point of the `fdfs` file system type. Normally, this mount is done automatically.

DIAGNOSTICS

`open(2)` returns –1 and `EBADF` if the associated file descriptor is not open.

SEE ALSO

`open(2)`, `dup(2)`.

System Administrator's Reference Manual

filehdr(4)

NAME

filehdr - file header for common object files

SYNOPSIS

```
#include <filehdr.h>
```

DESCRIPTION

Every common object file begins with a 20-byte header. The following C `struct` declaration is used:

```
struct filehdr
{
    unsigned short f_magic ;       /* magic number */
    unsigned short f_nscns ;       /* number of sections */
    long           f_timdat ;      /* time & date stamp */
    long           f_symptr ;      /* file ptr to symtab */
    long           f_nsyms ;       /* number of symtab entries */
    unsigned short f_opthdr ;      /* sizeof(opt and header) */
    unsigned short f_flags ;       /* flags */
};
```

`f_symptr` is the byte offset into the file at which the symbol table can be found. Its value can be used as the offset in `fseek(3S)` to position an I/O stream to the symbol table. The Reliant UNIX system optional header is 28 bytes. The valid magic numbers are given below:

```
#define I386MAGIC    0514   /* i386 Computer */
#define WE32MAGIC    0560   /* 3B2, 3B5, and 3B15 computers */
#define N3BMAGIC     0550   /* 3B20 computer */
#define NTVMAGIC     0551   /* 3B20 computer */
#define VAXWRMAGIC   0570   /* VAX writable text segments */
#define VAXROMAGIC   0575   /* VAX read only sharable text segments */
#define MIPSEBMAGIC  0x0160
#define MIPSELMAGIC  0x0162
#define SMIPSEBMAGIC 0x6001
#define SMIPSELMAGIC 0x6201
#define MIPSEBUMAGIC 0x0180
#define MIPSELUMAGIC 0x0182
```

The `f_timdat` value is taken from the `time(2)` system call. Flag bits for f-flags currently defined are:

```
#define F_RELFLG  0000001  /* relocation entries stripped */
#define F_EXEC    0000002  /* file is executable */
#define F_LNNO    0000004  /* line numbers stripped */
#define F_LSYMS   0000010  /* local symbols stripped */
#define F_MINIMAL 0000020
#define F_UPDATE  0000040
#define F_SWABD   0000100
#define F_PATCH   0002000
#define F_NODF    0002000
#define F_CPIC    0004000
#define F_PIX     0010000
```

SEE ALSO

`time(2)`, `fseek(3S)`, `a.out(4)`.

System Administrator's Reference Manual

fp_dptbl(4)

NAME

`fp_dptbl` - fixed-priority dispatcher parameter table

DESCRIPTION

The process scheduler (or dispatcher) is the portion of the kernel that controls allocation of the CPU to processes. The scheduler supports the notion of scheduling classes where each class defines a scheduling policy, used to schedule processes within that class. Associated with each scheduling class is a set of priority queues on which ready to run processes are linked. These priority queues are mapped by the system configuration into a set of global scheduling priorities which are available to processes within the class. (The dispatcher always selects for execution the process with the highest global scheduling priority that is eligible to run on a given CPU. See **PROCESSOR AFFINITY AND BINDING CONSIDERATIONS** below).

The priority queues associated with a given class are viewed by that class as a contiguous set of priority levels numbered from 0 (lowest priority) to *n* (highest priority – a configuration dependent value). The set of global scheduling priorities that the queues for a given class are mapped into might not start at zero and might not be contiguous (depending on the configuration).

The fixed-priority class maintains an in-core table, with an entry for each priority level, giving the properties of that level. This table is called the fixed-priority dispatcher parameter table (`fp_dptbl`). The `fp_dptbl` consists of an array of parameter structures (`struct fp_dpent`), one for each of the *n* priority levels. The properties of a given priority level *i* are specified by the *i*th parameter structure in this array (`fp_dptbl[`*i*`]`).

A parameter structure consists of the following members. These are also described in the `/usr/include/sys/fixpri.h` header file.

`fp_globpri`
: The global scheduling priority associated with this priority level. The mapping between fixed-priority priority levels and global scheduling priorities is determined at boot time by the system configuration. The `fp_globpri` values cannot be changed with `dispadmin`(1M).

`fp_quantum`
: The length of the time quantum allocated to processes at this level in ticks (HZ). The time quantum value is only a default or starting value for processes at a particular level as the time quantum of a fixed-priority process can be changed by the user with the `priocntl`(1) command or the `priocntl`(2) system call.

An administrator can affect the behavior of the fixed-priority portion of the scheduler by reconfiguring the `fp_dptbl`. There are two methods available for doing this.

KERNEL LINK TIME CONFIGURATION

The `fp_dptbl` can be reconfigured at kernel link time time by specifying the desired values in the `/etc/conf/pack.d/fp/space.c` file and reconfiguring the system using the auto-configuration procedure; see `idbuild`(1M). This is the only method that can be used to change the number of fixed-priority priority levels or the set of global scheduling priorities used by the fixed-priority class.

DISPADMIN CONFIGURATION FILE

The `fp_quantum` values in the `fp_dptbl` can be examined and modified on a running system using the `dispadmin`(1M) command. Invoking `dispadmin` for the fixed-priority class allows the administrator to retrieve the current `fp_dptbl` configuration from the kernel's in-core table, or overwrite the in-core table with values from a configuration file. The configuration file used for input to `dispadmin` must conform to the specific format described below.

Blank lines are ignored and any part of a line to the right of a `#` symbol is treated as a comment. The first non-blank, non-comment line must indicate the resolution to be used for interpreting the time quantum values. The resolution is specified as

`RES=`*res*

where *res* is a positive integer between 1 and 1,000,000,000 inclusive and the resolution used is the reciprocal of *res* in seconds. (For example, `RES=1000` specifies millisecond resolution.) Although very fine (nanosecond) resolution may be specified, the time quantum lengths are rounded up to the next integral multiple of the system clock's resolution.

The remaining lines in the file are used to specify the `fp_quantum` values for each of the fixed-priority priority levels. The first line specifies the quantum for fixed-priority level 0, the second line specifies the quantum for fixed-priority level 1, etc. There must be exactly one line for each configured fixed-priority priority level. Each `fp_quantum` entry must be a positive integer specifying the desired time quantum (in the resolution given by *res*).

EXAMPLE

The following excerpt from a `dispadmin` configuration file illustrates the format. Note that for each line specifying a time quantum there is a comment indicating the corresponding priority level. These level numbers indicate priority within the fixed-priority class, and the mapping between these fixed-priority priorities and the corresponding global scheduling priorities is determined by the configuration specified in the `fp/space.c` file. The level numbers are strictly for the convenience of the administrator reading the file and, as with any comment, they are ignored by `dispadmin` on input. `dispadmin` assumes that the lines in the file are ordered by consecutive, increasing priority level (from 0 to the maximum configured fixed-priority priority). The level numbers in the comments should normally agree with this ordering; if for some reason they don't, however, `dispadmin` is unaffected.

```
# Fixed Priority Dispatcher Configuration
RES=1000
# TIME QUANTUM                    PRIORITY
# (fp_quantum)                      LEVEL
      50                        #       0
      50                        #       1
      50                        #       2
      50                        #       3
      50                        #       4
      50                        #       5
      50                        #       6
      50                        #       7
      50                        #       8
      50                        #       9
      50                        #      10
      50                        #      11
       .                        .       .
       .                        .       .
       .                        .       .
      50                        #      73
      50                        #      74
```

PROCESSOR AFFINITY AND BINDING CONSIDERATIONS

In order to increase the cache hit rates of a multiprocessor, a process should generally execute on a particular processor. This affinity for a particular processor will reduce the required memory bandwidth for the process (and for the system as a whole). If a process has not run "recently" then it probably only has a small cache state in any given processor – and thus should loose its affinity for any particular processor.

This affinity mechanism can be enabled or disabled on a per-process basis by the super user [see `mpcntl(1M)`, `mpcntl(3X)`]. The affinity mechanism defaults to `on` for the FP class.

A process can also be bound to a set of processors. If a processor has been so bound, then it will *only* run on one of the processors of that set. If there is a idle processor that is not in the processes binding set – then the process will still wait until there is a processor in its set that is not running a higher priority process.

Process bindings can be established by the process [see `mpcntl(3X)`], or can be temporarily established by

the kernel (if the kernel is executing driver code that must run on a particular CPU).

FILES

`/usr/include/sys/fixpri.h`
`/etc/conf/pack.d/fp/space.c`

SEE ALSO

`priocntl`(1), `dispadmin`(1M), `idbuild`(1M), `priocntl`(2), `mpcntl`(3X).

System Administrator's Reference Manual

fs(4-ufs)

NAME

`fs(ufs)` - format of ufs file system volume

SYNOPSIS

```
#include <sys/param.h>

#include <sys/types.h>

#include <sys/fs/ufs_fs.h>
```

DESCRIPTION

Each disk drive contains some number of file systems. A file system consists of a number of cylinder groups. Each cylinder group has inodes and data.

A file system is described by its super-block, and by the information in the cylinder group blocks. The super-block is critical data and is replicated before each cylinder group block to protect against catastrophic loss. This is done at `mkfs` time; the critical super-block data does not change, so the copies need not normally be referenced further.

```
/*
 * Super-block for a file system.
 */
#define FS_MAGIC 0x011954
#define FSACTIVE 0x5e72d81a        /* fs_state: mounted */
#define FSOKAY   0x7c269d38        /* fs_state: clean */
#define FSBAD    0xcb096f43        /* fs_state: bad root */

struct fs
{
   struct  fs *fs_link;            /* linked list of file systems */
   struct  fs *fs_rlink;           /* used for incore super-blocks */
   daddr_t fs_sblkno;              /* addr of super-block in filesys */
   daddr_t fs_cblkno;              /* offset of cyl-block in filesys */
   daddr_t fs_iblkno;              /* offset of inode-blocks in filesys */
   daddr_t fs_dblkno;              /* offset of first data after cg */
   long    fs_cgoffset;            /* cylinder group offset in cylinder */
   long    fs_cgmask;              /* used to calc mod fs_ntrak */
   time_t  fs_time;                /* last time written */
   long    fs_size;                /* number of blocks in fs */
   long    fs_dsize;               /* number of data blocks in fs */
   long    fs_ncg;                 /* number of cylinder groups */
   long    fs_bsize;               /* size of basic blocks in fs */
   long    fs_fsize;               /* size of frag blocks in fs */
   long    fs_frag;                /* number of frags in a block in fs */
/* these are configuration parameters */
   long    fs_minfree;             /* minimum percentage of free blocks */
   long    fs_rotdelay;            /* num of ms for optimal next block */
   long    fs_rps;                 /* disk revolutions per second */
/* these fields can be computed from the others */
   long    fs_bmask;               /* "blkoff" calc of blk offsets */
   long    fs_fmask;               /* "fragoff" calc of frag offsets */
   long    fs_bshift;              /* "lblkno" calc of logical blkno */
   long    fs_fshift;              /* "numfrags" calc number of frags */
/* these are configuration parameters */
   long    fs_maxcontig;           /* max number of contiguous blks */
   long    fs_maxbpg;              /* max number of blks per cyl group */
/* these fields can be computed from the others */
```

```
   long    fs_fragshift;           /* block to frag shift */
   long    fs_fsbtodb;             /* fsbtodb and dbtofsb shift constant */
   long    fs_sbsize;              /* actual size of super-block */
   long    fs_csmask;              /* csum block offset */
   long    fs_csshift;             /* csum block number */
   long    fs_nindir;              /* value of NINDIR */
   long    fs_inopb;               /* value of INOPB */
   long    fs_nspf;                /* value of NSPF */
   long    fs_optim;               /* optimization preference, see below */
   long    fs_state;               /* file system state */
   long    fs_sparecon[2];         /* reserved for future constants */
/* a unique id for this filesystem (currently unused and unmaintained) */
   long    fs_id[2];               /* file system id */
/* sizes determined by number of cylinder groups and their sizes */
   daddr_t fs_csaddr;              /* blk addr of cyl grp summary area */
   long    fs_cssize;              /* size of cyl grp summary area */
   long    fs_cgsize;              /* cylinder group size */
/* these fields should be derived from the hardware */
   long    fs_ntrak;               /* tracks per cylinder */
   long    fs_nsect;               /* sectors per track */
   long    fs_spc;                 /* sectors per cylinder */
/* this comes from the disk driver partitioning */
   long    fs_ncyl;                /* cylinders in file system */
/* these fields can be computed from the others */
   long    fs_cpg;                 /* cylinders per group */
   long    fs_ipg;                 /* inodes per group */
   long    fs_fpg;                 /* blocks per group * fs_frag */
/* this data must be re-computed after crashes */
   struct  csum fs_cstotal;        /* cylinder summary information */
/* these fields are cleared at mount time */
  char    fs_fmod;                 /* super-block modified flag */
  char    fs_clean;                /* file system is clean flag */
  char    fs_ronly;                /* mounted read-only flag */
  char    fs_flags;                /* currently unused flag */
  char    fs_fsmnt[MAXMNTLEN];     /* name mounted on */
/* these fields retain the current block allocation info */
  long    fs_cgrotor;              /* last cg searched */
  struct  csum *fs_csp[MAXCSBUFS]; /* list of fs_cs info buffers */
  long    fs_cpc;                  /* cyl per cycle in postbl */
  short   fs_postbl[MAXCPG][NRPOS];/* head of blocks for each rotation */
  long    fs_magic;                /* magic number */
  u_char  fs_rotbl[1];             /* list of blocks for each rotation */
};

/*
 * Cylinder group block for a file system.
 */
#define CG_MAGIC 0x090255
struct cg
{
  struct cg *cg_link;          /* linked list of cyl groups */
  struct cg *cg_rlink;         /* used for incore cyl groups */
  time_t cg_time;              /* time last written */
  long   cg_cgx;               /* we are the cgx'th cylinder group */
  short  cg_ncyl;              /* number of cyl's this cg */
  short  cg_niblk;             /* number of inode blocks this cg */
  long   cg_ndblk;             /* number of data blocks this cg */
  struct csum cg_cs;           /* cylinder summary information */
  long   cg_rotor;             /* position of last used block */
  long   cg_frotor;            /* position of last used frag */
```

```
  long   cg_irotor;            /* position of last used inode */
  long   cg_frsum[MAXFRAG];    /* counts of available frags */
  long   cg_btot[MAXCPG];      /* block totals per cylinder */
  short  cg_b[MAXCPG][NRPOS];  /* positions of free blocks */
  char   cg_iused[MAXIPG/NBBY];/* used inode map */
  long   cg_magic;             /* magic number */
  u_char cg_free[1];           /* free block map */
};
```

SEE ALSO

inode(4-ufs) .

System Administrator's Reference Manual

fs(4-vxfs)
VxFS

NAME

`fs(vxfs)` - format of a Veritas File System volume (VxFS)

SYNOPSIS

```
#include <sys/types.h>
#include <sys/param.h>
#include <sys/fs/vx_fs.h>
```

DESCRIPTION

The Veritas File System (VxFS) super-block always begins at byte offset 1024 from the start of the file system. The super-block location is fixed so utilities know where to look for it.

The super-block contains the following fundamental sizes and offsets:

`fs_magic`
: The magic number for the file system (`VX_MAGIC`). This number identifies the file system as being a Veritas *FSType*.

`fs_version`
: The version number of the file system layout (`VX_VERSION`), currently 1.

`fs_ctime`
: The creation date of the file system. The `time(2)` system call supplies the time.

`fs_ectime`
: This field is a placeholder in instances when the creation date for a file system is expanded for more precision. It currently is zero.

`fs_obsolete1`
: Empty placeholder.

`fs_obsolete2`
: Empty placeholder.

`fs_logstart`
: The block address of the first Log Area block. It currently is 2.

`fs_logend`
: The block address of the last Log Area block. The Log Area size, in blocks, may be specified as part of `mkfs(1M)` . If not specified, a default of 256 blocks is used. A minimum size of 32 blocks is enforced.

`fs_bsize`
: The block size of the file system. The current choices are 1024, 2048, 4096, and 8192 bytes.

`fs_size`
: The number of blocks in the file system, expressed as the number of blocks of size `fs_bsize`. The `fs_size` field is a signed 32 bit number. The maximum number of blocks in a Veritas file system is limited to 31 bits.

`fs_dsize`
: The number of data blocks in the file system. A data block is a block which may be allocated to a file in the file system.

`fs_ninode`
: The number of inodes in the file system which is subject to the following rules:
  - The `fs_ninode` field is assigned 32 bit number.
  - The number of inodes is rounded down such that the inode list in each allocation unit is an integral number of inode list blocks (see `fs_ilbsize`).
  - The number of inodes may be specified as part of `mkfs(1M)` . If not specified, the default will be

fs_dsize divided by 4.

fs_nau
: The number of allocation units in the file system. The number of allocation units may be specified as part of mkfs(1M) . If not specified, the default will be fs_size rounded to the next highest 32K block boundary, divided by 32K.

fs_defiextsize
: The default size for indirect data extents, expressed in blocks. The default size for indirect data extents may be specified as part of mkfs(1M) . If not specified, a default of 64 blocks is used.

fs_ilbsize
: The size of an inode list block, expressed in bytes. This size may be specified as part of mkfs(1M) . If not specified, the maximum of 2KB or fs_bsize is used as the default.

fs_immedlen
: The size, in bytes, of the immediate data area in each inode. This value is 96 for the Veritas file system, version 1.

fs_ndaddr
: The number of direct extents supported by the VX_EXT4 mapping type (see the section describing inode list). This value is 10 for the Veritas file system version 1.

The preceding fields define the size and make up of the file system. To reduce the calculations required in utilities, a number of values are derived from the fundamental values and placed in the super-block.

The super-block contains the following derived offsets:

fs_aufirst
: The address, in blocks, of the first allocation unit. There can be a gap between the end of the intent log and the first allocation unit. This gap could be used to align the first allocation unit on a desired boundary.

fs_emap
: The offset, in blocks, of the free extent map (emap) from the start of an allocation unit.

fs_imap
: The offset, in blocks, of the free inode map (imap) from the start of an allocation unit.

fs_iextop
: The offset, in blocks, of the extended inode operation map from the start of an allocation unit.

fs_istart
: The offset, in blocks, of the inode list (ilist) from the start of an allocation unit.

fs_bstart
: The offset, in blocks, of the first data block from the start of an allocation unit. An allocation unit header may contain padding to align the first data block.

fs_femap
: The offset, in blocks, of the first free extent map (emap) from the start of the file system.

fs_fimap
: The offset, in blocks, of the first free inode map (imap) from the start of the file system.

fs_fiextop
: The offset, in blocks, of the first extended inode operation map from the start of the file system.

fs_fistart
: The offset, in blocks, of the first ilist from the start of the file system.

fs_fbstart
: The offset, in blocks, of the first data block from the start of the file system.

fs_nindir
: The number of entries in an indirect address extent. An indirect address extent is currently 8192 bytes in length, making the current value for fs_nindir 2048.

fs_aulen
: The length of an allocation unit in blocks.

`fs_auimlen`
: The length of a free inode map in blocks.

`fs_auemlen`
: The length of a free extent map in blocks.

`fs_ailen`
: The length, in blocks, of the inode list for this allocation unit.

`fs_aupad`
: The length, in blocks, of the allocation unit alignment padding.

`fs_ablocks`
: The number of data blocks in an allocation unit.

`fs_maxtier`
: The log base 2 of `fs_aublocks`.

`fs_inopb`
: The number of inode entries per `fs_bsize` block in the inode list.

`fs_inopau`
: The number of inodes in an allocation unit.

`fs_inopilb`
: The number of inode entries per `fs_ilbsize` block in the inode list.

`fs_ndiripau`
: Expected number of directory inodes per allocation unit.

`fs_iaddrlen`
: The size, in blocks, of an indirect address block. An indirect address block is 8K bytes. This field will be set to (8K/`fs_bsize`).

`fs_bshift`
: The log base 2 of `fs_bsize`. Used to convert a byte offset into a block offset.

`fs_inoshift`
: The log base 2 of `fs_inopb`. Used to convert an inode number into a block offset in the inode list.

`fs_bmask`
: A mask value such that (*byte_offset* and `fs_bmask`) rounds the offset to the nearest smaller block boundary.

`fs_boffmask`
: A mask value such that (*byte_offset* and `fs_boffmask`) yields the offset from the start of the nearest smaller block boundary.

`fs_inomask`
: A mask value such that (*inode_number* and `fs_inomask`) yields the offset from the start of the containing inode list block of the corresponding inode list entry.

`fs_checksum`
: A simple checksum of the above fields. A macro, `VX_FSCHECKSUM`, is provided to verify or calculate the checksum.

The above fields, which are initialized when the file system is created, do not change unless the file system is enlarged. These fields are replicated in each allocation unit header.

There are additional fields which are considered to be dynamic:

`fs_free`
: The current number of free data blocks.

`fs_ifree`
: The current number of free inodes.

`fs_efree`
: An array of the current number of free extents of each extent size in the file system.

`fs_flags`
: The following flags are recognized:

VX_FULLFSCK
: Set when a file system requires a full structural check to recover from an error. If this flag is set, a full check will be performed after the replay recovery is finished.

VX_NOLOG
: Set when the file system was mounted with the VX_MS_NOLOG option. If this flag is set, then no log replay recovery will be performed.

VX_LOGBAD
: Set when an I/O error has invalidated the log. If this flag is set, then no log replay recovery will be performed.

VX_LOGRESET
: Set when the log ID runs over VX_MAXLOGID (2^{30}). The log ID will be reset at the next appropriate opportunity (such as a mount or 60-second sync).

VX_RESIZE
: Set when a file system expansion is in progress. If an fsck(1M) sees this flag, it will have to perform resize recovery. Refer to fsadm(1M) for a description of file system expansion.

fs_mod
: Set whenever a mounted file system is modified. It is used to indicate if the super-block needs to be written when a sync operation is performed.

fs_clean
: Set to VX_DIRTY when a file system is mounted for read/write access. Set to VX_CLEAN upon umount(1M) or successful fsck(1M). The mount(1M) utility will not allow a file system to be mounted for read/write if the fs_clean field is VX_DIRTY.

fs_reserved
: Reserved for future use.

fs_firstlogid
: Starting intent log ID to use when the file system is next mounted.

fs_time
: Last time the super-block was written to disk. This field is in the same format as fs_ctime.

fs_etime
: Spare for when time becomes 8 bytes.

fs_fname
: File system name (6 characters).

fs_fpack
: File system pack label (6 characters).

SEE ALSO

fsck(1M), fsdb(1M), mkfs(1M), mount(2), inode(4-vxfs).

fspec(4)

NAME

`fspec` - format specification in text files

DESCRIPTION

It is sometimes convenient to maintain text files on the Reliant UNIX system with non-standard tabs, (i.e. tabs that are not set at every eighth column). Such files must generally be converted to a standard format, frequently by replacing all tabs with the appropriate number of spaces, before they can be processed by Reliant UNIX system commands. A format entry in the first line of a text file specifies how tabs are to be expanded in the remainder of the file.

A format specification consists of a sequence of parameters separated by blanks and surrounded by the brackets `<:` and `:>`. Each parameter consists of a keyletter, possibly followed immediately by a value. The following parameters are recognized:

`t`*tabs*
: The `t` parameter specifies the tab settings for the file. The value of `tabs` must be one of the following:

  1. a list of column numbers separated by commas, indicating tabs set at the specified column;
  2. a – followed immediately by an integer *n*, indicating tabs at intervals of *n* columns;
  3. a – followed by the name of a "canned" tab specification.

  Standard tabs are specified by `t-8`, or equivalently, `t1,9,17,25`, etc. The canned tabs that are recognized are defined by the `tabs(1)` command.

`s`*size*
: The `s` parameter specifies a maximum line size. The value of *size* must be an integer. Size checking is performed after tabs have been expanded, but before the margin is prepended.

`m`*margin*
: The `m` parameter specifies a number of spaces to be prepended to each line. The value of *margin* must be an integer.

`d`
: The `d` parameter takes no value. Its presence indicates that the line containing the format specification is to be deleted from the converted file.

`e`
: The `e` parameter takes no value. Its presence indicates that the current format is to prevail only until another format specification is encountered in the file.

Default values, which are assumed for parameters not supplied, are `t-8` and `m0`. If the `s` parameter is not specified, no size checking is performed. If the first line of a file does not contain a format specification, the above defaults are assumed for the entire file. The following is an example of a line containing a format specification:

```
* <:t5,10,15 s72:> *
```

If a format specification can be disguised as a comment, it is not necessary to code the `d` parameter.

SEE ALSO

`ed(1)`, `newform(1)`, `tabs(1)`.

gettydefs(4) - issue(4)

gettydefs(4)

NAME

`gettydefs` - speed and terminal settings used by getty

DESCRIPTION

The /etc/gettydefs file contains information used by getty(1M) to set up the speed and terminal settings for a line. It supplies information on what the login prompt should look like. It also supplies the speed to be tried next if the user indicates the current speed is not correct by pressing the BREAK key.

Each entry in /etc/gettydefs has the following format:

label# *initial-flags* # *final-flags* # *login-prompt* #*next-label*

Each entry is followed by a blank line. Lines that begin with # are ignored and may be used to comment the file. The various fields can contain quoted characters of the form \b, \n, \c, etc. as well as *nnn*, where *nnn* is the octal value of the desired character. The various fields are:

label
: This is the string against which getty(1M) tries to match its second argument. It is often the speed, such as 1200, at which the terminal is supposed to run, but it needn't be (see below).

initial-flags
: These flags are the initial ioctl(2) settings to which the terminal is to be set if a terminal type is not specified to getty(1M). getty(1M) understands the symbolic names specified in /usr/include/sys/termio.h [see termio(7)]. Normally only the speed flag is required in the *initial-flags*. getty(1M) automatically sets the terminal to raw input mode and takes care of most of the other flags. The \flinitial-flag settings remain in effect until getty(1M) executes login(1).

final-flags
: These flags take the same values as the *initial-flags* and are set just prior to getty(1M) executing login. The speed flag is again required. The composite flag SANE takes care of most of the other flags that need to be set so that the processor and terminal are communicating in a rational fashion. The other two commonly specified *final-flags* are TAB3, so that tabs are sent to the terminal as spaces, and HUPCL, so that the line is hung up on the final close.

login-prompt
: This entire field is printed as the \fllogin-prompt. Unlike the above fields where white space is ignored (a space, tab or new-line), they are included in the *login-prompt* field.

next-label
: This indicates the next *label* of the entry in the table that getty(1M) should use if the user presses the BREAK key or the input cannot be read. Usually, a series of speeds are linked together in this fashion, into a closed set. For instance, 2400 linked to 1200, which in turn is linked to 300, which finally is linked to 2400.

If getty(1M) is called without a second argument, then the first entry of /etc/gettydefs is used, thus making the first entry of /etc/gettydefs the default entry. It is also used if getty(1M) can't find the specified *label*. If /etc/gettydefs itself is missing, there is one entry built into the command which will bring up a terminal at 300 baud.

After making or modifying /etc/gettydefs, it is strongly recommended that it be run through getty(1M) with the check option to make sure there are no errors.

FILES

/etc/gettydefs

SEE ALSO

login(1), getty(1M), ioctl(2), termio(7).

System Administrator's Reference Manual

group(4)

NAME

`group` - group file

DESCRIPTION

The `/etc/group` file contains for each group the following information:

- group name
- encrypted password
- numerical group ID
- comma-separated list of all users allowed in the group

`group` is an ASCII file. The fields are separated by colons; each group is separated from the next by a new line.

Because of the encrypted passwords, the group file can and does have general read permission and can be used, for example, to map numerical group IDs to names.

During user identification and authentication, the supplementary group access list is initialized sequentially from information in this file. If a user is in more groups than the system is configured for [see `NGROUPS_MAX` in `limits(4)`], a warning will be given and subsequent group specifications will be ignored.

SEE ALSO

`groups(1)`, `newgrp(1)`, `passwd(1)`, `setgroups(2)`, `initgroups(3C)`, `limits(4)`, `unistd(4)`.

System Administrator's Reference Manual

hdlcdefs(4)

NAME

`hdlcdefs` - timer and counter settings for HDLC lines

SYNOPSIS

`/etc/hdlcdefs`

DESCRIPTION

The `hdlcdefs` file contains information on the line settings for HDLC connections. Every line has the following format:

`link`:*label*:*sections*
determines the HDLC parameters for a device address;

or

`line`:*label*:*sections*
determines the HDLC parameters for all devices in a line.

label is a freely selectable flag that indicates the device- or line-related HDLC parameters. *label* is referenced by `hdlclabs(4)` or the relevant monitor, for example.

sections can be either a list of individual terms separated by blanks, or tokens followed by an integer *value*. The following options are possible:

1) `link`-specific parameters

`blocked`/`unblocked`
The values `blocked` or `unblocked` are permitted. `blocked` means that `write`/`read` is blocked; blocked mode is activated when the application takes control of a HDLC frame via the data field. `unblocked` sets the default mode.

Default: `unblocked`.

`framelength`=*value*
Length of the send data. The application uses this parameter to set the maximum length of a frame. A length of 1 kbyte is the maximum value *value* permitted (the maximum value for `SIhmc` is 4 kbytes, minimum value 64 bytes). If a `write` command is issued with *n* data, the `write` can only be executed correctly if *n* is less than or equal to the maximum size of the send frame.

If a value >1 KByte is set using `ioctl` for `framelength`, the `ioctl` is rejected with the error `EINVAL`.

Default: 100.

`actpcnt`=*value*
Active Poll Count. Specifies how many poll attempts are to be made to send or receive an I-frame without delay. The counter is reset every time an I-frame is sent or received.

When the timer expires, the primary host for this terminal switches to "inactive" mode.

Default: 2350 (this means a period of approx. 2 minutes with a polling frequency of 50 ms).

`iactpoll`=*value*
Active Poll. Polling frequency for fast polls. Specifies the frequency with which a terminal in active mode can be polled.

Default: 50 ms.

`iinactpoll`=*value*
Inactive Poll. Polling frequency for delayed polls. Specifies the frequency with which a terminal in inactive mode is to be polled.

A terminal is said to be inactive if the repeat counter for `actpcnt` has expired.

Default: 500 ms.

`inoresppoll`=*value*
: No Response Poll. Polling frequency for terminals in "no response" mode. Specifies the frequency with which a terminal in "no response" mode is to be polled.

A terminal is said to be in "no response" mode if a long period of time has elapsed since it answered a poll. The terminal is probably switched off or not connected.

Default: 10000 ms.

`istdbypoll`=*value*
: Standby Poll. Polling frequency for terminals in standby poll mode. Specifies the frequency with which a terminal in standby poll mode is to be polled.

A terminal is said to be in standby poll mode if an `open` exists but the terminal has not yet answered.

Default: 5000 ms.

`m5timer`=*value*
: Specifies the maximum time the last secondary polled will wait before canceling the `S2` signal. After the timer has expired, the next secondary is polled. If `S2` is not canceled after the specified time has elapsed, the next secondary is nonetheless polled.

Default: 10 ms.

`resptimer`=*value*
: Response timer in normal response mode (NRM). Specifies the maximum waiting time for a response in NRM if an I-frame is expected. The last job is repeated after the timer expires.

Default: 300 ms.

`resptocnt`=*value*
: Response Timeout Count. Specifies how often a job is repeated if the terminal does not respond. This parameter is no longer evaluated. The `rptcnt` counter is used.

Default: 6.

`rptcnt`=*value*
: Repeat Count. Repeat counter for I-frames. Specifies how often an unacknowledged I-frame is repeated provided that the terminal has responded.

When this counter expires, an error message is generated and the terminal switches to disconnected mode.

Default: 6.

`snrmresptimer`=*value*
: Set Normal Response Mode Timer. Specifies the maximum waiting time for a quick response (e.g. UA). The job is repeated after the timer has expired.

Default: 32 ms.

2) `line`-specific parameters

`twowire`/`fourwire`
: Select the line type twowire or fourwire. Specifies whether the line is a four-wire or two-wire line. If it is a two-line wire, the `S2` signal must be canceled after every frame is sent; this does not apply to a four-wire line.

For a two-wire line, set the line type to `twowire`.

For a four-wire line (e.g. for a ba80 terminal), set the line type to `fourwire`.

This parameter only refers to the physical operation of the line. Logically, half-duplex mode is always used.

Default: `fourwire`.

`chartimer`=*value*
: Character Timer. Specifies the maximum time for sending and receiving a frame. This means that the process of sending and receiving the entire frame is monitored. The character timer depends on the line speed and the length of the frame. The timer should be set to cover all speeds. If a frame cannot

be sent or received within the specified time, the terminal switches to error status.

Default: 5000 ms.

`m2timer=`*value*

`M2` monitoring timer. Specifies the maximum waiting time for the `M2` message.

If `M2` was not set within this time, the frame is not sent and the terminal switches to error status.

Default: 3000 ms.

`s2timer=`*value*

`S2` monitoring timer. Specifies the maximum waiting time for data to be sent over the physical line after the `S2` signal has been issued.

Default: 2 s.

`speed=`*value*

Specifies the speed (not for `SIsih` and `sihmon`). *value* is a positive number that can precede the uppercase or lowercase letter `K`. If the number ends with uppercase or lowercase `K`, it is a multiple of 1024.

The parameter `speed` is not evaluated for `SIsih` and `sihmon`. The speed in this case is determined from the label used from `/etc/ttydefs`.

The HDLC monitor searches `/etc/hdlcdefs` until the first suitable flag is found.

NOTES

Lines beginning with `#` are ignored; they can therefore be used as comment lines.

The default entries (*label*=`default`) are generated by the `SIterms` package.

EXAMPLES

```
link:default:actpcnt=2350 rptcnt=6 resptocnt=6
link:li3:resptimer=350
line:default:twowire blocked
```

FILES

`/etc/hdlcdefs`

SEE ALSO

`hdlclabs(4)`.

System Administrator's Reference Manual

hdlclabs(4)

NAME

`hdlclabs -` settings for HDLC line parameters

SYNOPSIS

`/var/sadm/terms/hdlclabs`

DESCRIPTION

The `hdlclabs` file contains information on the line settings for HDLC devices. Each line has the following format:

line-label:*hdcldefs-label*

For every HDLC *line* there is a corresponding entry which specifies the *label* in the `hdlcdefs` file (*hdlcdefs-label*); the actual line settings for the relevant line are stored here.

The entries in this file are created by `termadd` after the first device in a line is configured; they are deleted again by `termdel` when deleting the last device in a string.

line-label is composed of the controller type ("sih") as well as the line number. The lines are numbered from 0 onwards. There are separate numbers for each type of controller.

EXAMPLES

```
sih0:default
sih3:testline
```

SEE ALSO

`termadd`(1M), `termdel`(1M).

System Administrator's Reference Manual

idevice(4)
RM400 only

NAME

`idevice -` interrupt configuration file

SYNOPSIS

`/etc/conf/cf.d/idevice`

DESCRIPTION

Each package that uses EISA/ISA bus interrupts must supply an `idevice` file which specifies the interrupt requirements of the package.

When the package is installed (using `idinstall`), the `idevice` part of the package is copied into the `idevice.d` directory of the kernel link kit. At kernel link time, these component files are concatenated into a single file which is then processed by `idintconf`. This command generates a data structure that is used at kernel boot time to allocate the use of EISA/ISA bus interrupt resources.

The `idevice` file consists of one or more lines, each line describes an interrupt resource required by the device. Each line consists of *nine* white space separated fields. The first field can be quoted using double quotes. The other fields cannot be quoted. All fields must be supplied. Comment lines are lines that start with a "`#`" or a "`*`".

Where allowed a "`–`" character is used to specify that the field should be undefined. The "`*`" character is used to specify the default value for the field. Normally this consists of the `Internal name` with a field dependent string appended to it.

In later revisions of the file, it is expected that more fields will be added to each line. It is thus inadvisable to place comments at the end of the line.

The current implementation contains the following fields:

`Colloquial name`
: The first field of each line is a human readable name for the device which is used when `idintconf` or the kernel must generate a diagnostic. This field must be unique for all lines relating to the same driver. This field can be quoted using double quotes. It is the only field that can be quoted.

`Internal name`
: The second field is the name to be used to generate symbols for the kernel. It is the same as the `Device name` field in `sdevice`. This field is used to generate other fields that have been defaulted.

  If this field is set to "`–`" or "`*`", the value of this field is the same as the `Colloquial name`. This is not allowed if the `Colloquial name` contains any white space characters.

`Bus type`
: This field specifies which type of bus interface the hardware has. This can either be ISA or EISA. There is no default for this field. This field can be shortened to "I" or "E" respectively.

`Interrupt type`
: If the hardware has an ISA interface, this field can be set to `EDGE` or a "`–`" which defaults to `EDGE`. No other value can be used for ISA cards. This is because of a hardware restriction in the original design of the AT bus.

  If the hardware has an EISA interface, this field is used to specify the type of interrupt that the hardware can handle. This can be either `EDGE`, for edge triggered devices, or `LEVEL` for level triggered devices, or `SHARED`. There is no default for this field on an EISA bus device.

  The `SHARED` type can be used for devices which use level triggered interrupts and which can determine at interrupt time if the hardware has actually interrupted. This is because all interrupt routines that share an interrupt will be called when an interrupt is detected. The order in which these interrupt routines is called is NOT defined.

This field can be shortened to the first letter of the value if required.

`Number of Interrupts`
Most devices usually use only one interrupt for each card so this field is normally set to 1. However some cards use more than one interrupt per hardware card, e.g. the `chase` card. In this case, this field is set to the number of interrupts that each card uses. For example the `chase` would have a value of 2.

There is no default value for this field.

`Possible Interrupts`
This field consists of a comma separated list of possible interrupts that the card can use. This list is logically grouped into n-tuples. The size of each n-tuple is defined by the `Number of interrupts` field. In the normal case the n-tuple has a length of 1 so that each value is treated separately.

For ISA bus devices, the interrupt number cannot be changed under software control (on some cards this may be possible but in general it is not). This means that only the first n-tuple is used. `idintconf` and the kernel will give warnings if there is a clash of interrupt numbers. In this case the following n-tuples are used to inform the user of a possible jumpering change that would allow the use of the card. If there is an interrupt clash between two ISA bus devices, both devices will be disabled. This is because of a hardware restriction in the original AT bus design.

On EISA bus devices, the interrupt number can be dynamically assigned so as not to conflict with other devices. In this case each n-tuple is tried in turn. If the n-tuple does not conflict with any other interrupts, then the card is assigned that n-tuple. If however all n-tuples cause a conflict, the kernel tries to rearrange all conflicting interrupts so that they no longer clash. If two or more devices are currently sharing an interrupt, then they will not be rearranged. In this way, it is hoped that all interrupt conflicts can be avoided. If the conflicting device is an ISA device, its interrupt number cannot be changed. If all possible arrangements of interrupts cause conflicts, then the device will not be enabled, and no interrupt will be assigned to the device.

The n-tuples of this field should be arranged in preferred order, i.e. the interrupt number that is most preferred should be the first on the list.

`Interrupt routine`
This field specifies the name of the interrupt handling function to be used by this device. In the normal case, where the driver only uses one interrupt per device, this field can be defaulted by supplying a "`*`". If this is done, the function will be called *internal_name*`intr`.

If the driver requires more than one interrupt per device, this field will then become a comma separated list of interrupt functions that are to be used. In this case there is no default action since enough interrupt routines per device must be supplied. The ordering with respect to interrupt numbers is lexical. I.e. the first interrupt routine is called by an interrupt on the first interrupt number, the second interrupt routine is called by an interrupt on the second interrupt number.

If a "–" is supplied, then there is no interrupt function specified, the use of this is for future study.

`Existence routine`
This field is the name of a routine which is the first one of the driver called. It is called before the drivers' `init` routine. The purpose of this routine is to determine if the device actually exists. If the device exists, then the routine should return `PROBE_OK`. If the device does not exist, the routine should return `PROBE_FAIL`. If the driver cannot determine if the device exists, it should return `PROBE_OK`.

To allow the driver to determine which hardware device is to be checked, this routine is passed the address of a `intr_param_t` structure. The fields in this structure can be used to determine which device is being checked.

If this field is a "–", then no existence routine is supplied, in this case the kernel will assume that the device exists and proceed as if the routine had returned `PROBE_OK`.

If this field is a "`*`", the name of the routine will be *internal_name*`exist`.

For more information, see the `XXexist(5)` manual page.

`Set interrupt routine`
This field specifies the name of a function that is called after the `exist` routine but before the `init`

routine. This function is called if the kernel has been able to allocate interrupts for the device. If this routine is not called, the kernel was not able to allocate interrupts for the device and so the device should not initialize itself. A driver which returns `PROBE_FAIL` from the `exist` routine for a given device will never have its `setint` routine called for that device.

This routine is passed two parameters, the first is a pointer to a list of integers which are the interrupt lines that have be assigned to the given device. A pointer is used because some devices require more than one interrupt. It is assumed that the setint routine knows how many interrupts it has been assigned.

The second parameter is a pointer to a `intr_param_t` structure, the members of this structure can be used to determine which device the interrupts are being assigned to.

If a "`*`" is given in this field, the routine will be called *internal_name*`setint`. If a "–" is given, then no `setint` function has been supplied, the use of this value is for future study.

If neither the `setint` or `exist` routines is called, the driver can assume that it is running on a system which does not have dynamic interrupt assignment i.e. a pre 5.42 kernel.

Use of idevice

Each line in the `idevice` file describes an instance of a driver/device pair. Each line in the `idevice` file must have a corresponding entry in the `sdevice` file. Each line in the `idevice` defines the requirements of one piece of hardware. If a driver can support multiple EISA devices that can share resources i.e. interrupt numbers and interrupt devices, then only one line need be supplied in `idevice`. In this case the `sdevice` entry will enable or disable all of the devices. And it is the responsibility of the driver to handle the `exist` and `setint` routines correctly.

FILES

`/etc/conf/cf.d/sdevice`

SEE ALSO

`idintconf`(1M), `XXexist`(5), `XXsetint`(5).

System Administrator's Reference Manual

init(4)

NAME

`init` - /etc/default/init file

DESCRIPTION

The `/etc/default/init` file contains the default settings for the `init`(1M) command.

FORMAT

This format of this file is *identifier=value* where both the identifiers and values are defined by the `init`(1M) program.

CONTENTS

`ULIMIT`

The value is either a number or the string `unlimited`. This is used to set the initial ulimit for all processes on the system. The ulimit may be changed by programs such as `login`(1). This is especially useful for the start-up scripts.

DEFAULT VALUES

`ULIMIT=32768`

SEE ALSO

`login`(1), `init`(1M).

System Administrator's Reference Manual

inittab(4)

NAME

`inittab` - script for init

DESCRIPTION

The file `/etc/inittab` controls process dispatching by `init`. The processes most typically dispatched by `init` are daemons.

The `inittab` file contains position-dependent entries with the following format:

id : *rstate* : *action* : *process*

Each entry is delimited by a newline, however a backslash (\) preceding a newline indicates a continuation of the entry. Up to 512 characters per entry are permitted. Comments may be inserted in the *process* field using the # character.

The entry fields are:

id This is one or two characters used to uniquely identify an entry.

rstate

This defines the run level in which this entry is to be processed. Run levels effectively correspond to a configuration of processes in the system. That is, each process spawned by `init` is assigned a run level or run levels in which it is allowed to exist. The run levels are represented by a number ranging from `0` through `6`. As an example, if the system is in run level `1`, only those entries having a `1` in the *rstate* field are processed. When `init` is requested to change run levels, all processes that do not have an entry in the *rstate* field for the target run level are sent the warning signal `SIGTERM` and allowed a 5-second grace period before being forcibly terminated by the kill signal `SIGKILL`. The *rstate* field can define multiple run levels for a process by selecting more than one run level in any combination from `0` through `6`. If no run level is specified, then the process is assumed to be valid at all run levels `0` through `6`. There are three other values, `a`, `b` and `c`, which can appear in the *rstate* field, even though they are not true run levels. Entries which have these characters in the *rstate* field are processed only when an `init` or `telinit` process requests them to be run (regardless of the current run level of the system), [see `init`(1M)]. They differ from run levels in that `init` can never enter run level `a`, `b` or `c`. Also, a request for the execution of any of these processes does not change the current run level. Furthermore, a process started by an `a`, `b` or `c` command is not killed when `init` changes levels. They are killed only if their line in `inittab` is marked `off` in the *action* field, their line is deleted entirely from `inittab`, or `init` goes into singleuser state.

action

Key words in this field tell `init` how to treat the process specified in the *process* field. The actions recognized by `init` are as follows:

`respawn`
If the process does not exist, then start the process; do not wait for its termination (continue scanning the `inittab` file), and when the process dies, restart the process. If the process currently exists, do nothing and continue scanning the `inittab` file.

`wait` When `init` enters the run level that matches the entry's *rstate*, start the process and wait for its termination. All subsequent reads of the `inittab` file while `init` is in the same run level cause `init` to ignore this entry.

`once` When `init` enters a run level that matches the entry's *rstate*, start the process, do not wait for its termination. When it dies, do not restart the process. If `init` enters a new run level and the process is still running from a previous run level change, the program is not restarted.

`boot` The entry is to be processed only at `init`'s boot-time read of the `inittab` file. `init` is to start the process, not wait for its termination; and when it dies, not restart the process. In order for this instruction to be meaningful, the *rstate* should be the default or it must match `init`'s run level at

boot time. This action is useful for an initialization function following a hardware reboot of the system.

`bootwait`
The entry is to be processed the first time `init` goes from singleuser to multiuser state after the system is booted. (If `initdefault` is set to `2`, the process runs right after the boot.) `init` starts the process, waits for its termination and, when it dies, does not restart the process.

`powerfail`
Execute the process associated with this entry only when `init` receives a power fail signal, `SIGPWR` [see `signal(2)`].

`powerwait`
Execute the process associated with this entry only when `init` receives a power fail signal, `SIGPWR`, and wait until it terminates before continuing any processing of `inittab`.

`off` If the process associated with this entry is currently running, send the warning signal `SIGTERM` and wait 5 seconds before forcibly terminating the process with the kill signal `SIGKILL`. If the process is nonexistent, ignore the entry.

`ondemand`
This instruction is really a synonym for the `respawn` action. It is functionally identical to `respawn` but is given a different keyword in order to divorce its association with run levels. This instruction is used only with the `a`, `b` or `c` values described in the *rstate* field.

`initdefault`
An entry with this action is scanned only when `init` is initially invoked. `init` uses this entry, if it exists, to determine which run level to enter initially. It does this by taking the highest run level specified in the *rstate* field and using that as its initial state. If the *rstate* field is empty, this is interpreted as `0123456` and `init` therefore enters run level `0`. This will cause the system to stop. Additionally, if `init` does not find an `initdefault` entry in `inittab`, it requests an initial run level from the user at reboot time.

`sysinit`
Entries of this type are executed before `init` tries to access the console (i.e. before the `Console Login:` prompt). It is expected that this entry will be only used to initialize devices on which `init` might try to ask the run level question. These entries are executed and waited for before continuing.

process
This is a command to be executed. The entire *process* field is prefixed with `exec` and passed to a forked `sh` as `sh -c 'exec command'`. For this reason, any legal `sh` syntax can appear in the *process* field.

Comment can be appended to the process entry and is introduced with `#`.

A maximum of 2,500 processes can be entered.

SEE ALSO

`who(1)`, `init(1M)`, `ttymon(1M)`, `exec(2)`, `open(2)`, `signal(2)`.

System Administrator's Reference Manual

inode(4-ufs)

NAME

`inode(ufs)` - format of a ufs inode

SYNOPSIS

```
#include <sys/param.h>
#include <sys/types.h>
#include <sys/vnode.h>
#include <sys/fs/ufs_inode.h>
```

DESCRIPTION

The inode is the focus of all local file activity in Reliant UNIX. There is a unique inode allocated for each active file, each current directory, each mounted-on file, each mapping, and the root. An inode is "named" by its dev/inumber pair. Data in `icommon` is read in from the permanent inode on the actual volume.

```
#define EFT_MAGIC 0x90909090    /* magic cookie for EFT */
#define NDADDR    12            /* direct addresses in inode */
#define NIADDR    3             /* indirect addresses in inode */
struct inode
{
    struct   inode *i_chain[2];/* must be first */
    struct   vnode i_vnode;    /* vnode associated with this inode */
    struct   vnode *i_devvp;   /* vnode for block I/O */
    u_short  i_flag;
    dev_t    i_dev;            /* device where inode resides */
    ino_t    i_number;         /* i number, 1-to-1 with device address */
    off_t    i_diroff;         /* offset in dir, where we found last entry */
    struct   fs *i_fs;         /* file sys associated with this inode */
    struct   dquot *i_dquot;   /* quota structure controlling this file */
    short    i_owner;          /* proc index of process locking inode */
    short    i_count;          /* number of inode locks for i_owner */
    short    i_rwowner;        /* proc index of process holding rwlock */
    daddr_t  i_nextr;          /* next byte read offset (read-ahead) */
    struct inode  *i_freef;    /* free list forward */
    struct inode **i_freeb;    /* free list back */
    ulong    i_vcode;          /* version code attribute */
    ulong    i_mapcnt;         /* mappings to file pages */
    int      *i_map;           /* block list for the corresponding file */
    struct   icommon
    {
        o_mode_t ic_smode;     /* 0: mode and type of file */
        short    ic_nlink;     /* 2: number of links to file */
        o_uid_t  ic_suid;      /* 4: owner's user id */
        o_gid_t  ic_sgid;      /* 6: owner's group id */
        quad     ic_size;      /* 8: number of bytes in file */
#ifdef _KERNEL
        struct timeval ic_atime;  /* 16: time last accessed */
        struct timeval ic_mtime;  /* 24: time last modified */
        struct timeval ic_ctime;  /* 32: last time inode changed */
#else
        time_t  ic_atime;      /* 16: time last accessed */
        long    ic_atspare;
        time_t  ic_mtime;      /* 24: time last modified */
        long    ic_mtspare;
        time_t  ic_ctime;      /* 32: last time inode changed */
```

```
        long    ic_ctspare;
#endif
        daddr_t ic_db[NDADDR]; /* 40: disk block addresses */
        daddr_t ic_ib[NIADDR]; /* 88: indirect blocks */
        long    ic_flags;      /* 100: status, currently unused */
        long    ic_blocks;     /* 104: blocks actually held */
        long    ic_gen;        /* 108: generation number */
        mode_t  ic_mode;       /* 112: EFT version of mode*/
        uid_t   ic_uid;        /* 116: EFT version of uid */
        gid_t   ic_gid;        /* 120: EFT version of gid */
        ulong   ic_eftflag;    /* 124: indicate EFT version*/
    } i_ic;
};
struct dinode
{
    union
    {
 struct icommon di_icom;
 char   di_size[128];
    } di_un;
};
```

SEE ALSO

fs(4-ufs).

System Administrator's Reference Manual

inode(4-vxfs)
VxFS

NAME

`inode(vxfs)` - format of a Veritas File System inode (VxFS)

SYNOPSIS

```
#include <sys/types.h>
#include <sys/fs/vx_inode.h>
```

DESCRIPTION

The inode list consists of `fs_inopau` inode entries. An inode entry has the following format:

`i_mode`
: The mode and type of file.

`i_nlink`
: The number of links to the file.

`i_uid`
: The inode owner.

`i_gid`
: The inode group.

`i_size`
: The size in bytes of the file. Eight bytes have been allocated. Only four bytes are used in the first implementation.

`i_atime`
: Time of last access, in `time_t` format.

`i_atspare`
: Reserved for when `i_atime` becomes eight bytes.

`i_mtime`
: Time of last modification, in `time_t` format.

`i_mtspare`
: Reserved for when `i_mtime` becomes eight bytes.

`i_ctime`
: Time of last inode change, in `time_t` format.

`i_ctspare`
: Reserved for when `i_ctime` becomes eight bytes.

`i_aflags`
: These flags are used to control the allocation and extension of files.

  `VX_AF_IBAD`
  : If this flag is set, the inode is invalid in some way. It should be cleared when `fsck(1M)` is run.

  `VX_AF_NOEXTEND`
  : If this flag is set, the file may not be extended once the current reservation is exceeded. The reservation may be increased by the `VX_SETEXT ioctl`, but the file will not be extended automatically.

  `VX_AF_ALIGN`
  : If this flag is set, the file must be allocated in extents of a fixed size and alignment. If an extent of `i_fixextsize` blocks aligned on an `i_fixextsize` boundary cannot be found, then the allocation will fail. The alignment is relative to the beginning of the allocation unit.

  `VX_AF_NOGROW`
  : If this flag is set, the file may not be extended once the current reservation is exceeded. It should be cleared on truncation or when `setext(1)` is run. This flag is usually set because an I/O

error occurs while extending a file.

`i_orgtype`
Mapping type. Indicates how the inode mapping area is to be interpreted. Currently there are two mapping types supported:

`IORG_EXT4`
Mapping area consists of an array of 32-bit extent block addresses and sizes.

`IORG_IMMED`
Mapping area itself is a data block. This mapping is referred to as Immediate Inode Data.

`i_eopflags`
Extended inode operation flag area.

`i_eopdata`
Extended inode operation data area.

`i_ftarea`
This is a union. The contents are determined by file type.

For devices, the following field is supported:

`i_rdev`
The device number of a block or character special device.

For directories, the following field is supported:

`i_dotdot`
The parent directory inode number if the inode is a directory. This replaces the standard ".." entry in the first directory block. The Veritas file system does not have explicit "." and ".." entries.

For regular files, the following fields are supported:

`i_reserve`
The number of data blocks reserved for exclusive use by the file (preallocation). A preallocation may be requested using `ioctl(2)`.

`i_fixextsize`
Set when the inode has a fixed extent size. The default is to have a variable extent size allocation policy. A fixed extent size may be specified using `ioctl(2)` [see `vxfsio(7)`].

`i_blocks`
The number of blocks currently allocated to the file, including any blocks allocated for indirect address extents.

`i_gen`
The generation number. A serial number which is incremented whenever the inode is freed and reallocated. It is designed to provide a "handle" for stateless servers, such as NFS.

`i_serial`
A count of the number of times the inode metadata has been modified. This field is a 64-bit number.

`ic_org`
The mapping area. This field is a union based on the value of `i_orgtype` and the file system type.

For the VxFS `IORG_IMMED` organization type, the following structure is used:

`i_immed`
The immediate inode data area, `NIMMED_N` (currently 96) bytes in length (see `fs_immedlen`). Any directory or symbolic link which is ≤ 96 bytes in length will be stored directly in the inode.

For the VxFS `IORG_EXT4` organization type the following structure is used:

`i_spare`
Four bytes of padding, not used.

`i_ies`
Indirect extent size. This is the size in bytes of the indirect data extents in the file.

`i_ie` Array of indirect address extents. There are `NIADDR` (currently 2) indirect address extents. The

indirect address extents are 8192 bytes long. Each indirect address extent may contain up to 2048 extent addresses. The first indirect address extent is used for single indirection. With single indirection, each entry in the indirect address extent indicates the starting block number of a data extent. The second indirect address extent is a double indirect address extent. With double indirection, each entry in the indirect address extent indicates the starting block number of a single indirect address extent.

`i_dext`
An array of structures containing the direct extent addresses and sizes. Up to `NDADDR_N` (currently ten) direct extents are supported. Since a variable length extent allocation policy is used, each direct extent may have a different size. Each structure contains the following elements:

`i_de` Direct extent address.

`i_des`
Direct extent size.

`reserved`
There are 80 bytes reserved for future use.

SEE ALSO

`stat`(2), `fs`(4-vxfs), `types`(5).

System Administrator's Reference Manual

issue(4)

NAME

`issue` - issue identification file

SYNOPSIS

`/etc/issue`

DESCRIPTION

The `/etc/issue` file contains the issue or project identification to be printed as a login prompt. `issue` is an ASCII file that is read by the programs `getty` and `ttymon` and then written to any terminal spawned or respawned.

SEE ALSO

`login(1)`.

LC_TIME(4) - ott(4)

System Administrator's Reference Manual

LC_TIME(4)

NAME

`LC_TIME` - language specific time and date strings

SYNOPSIS

`/usr/lib/locale/` *locale*`/LC_TIME`

DESCRIPTION

A readable file exists for each local environment which contains information on the respective date and time formats. This file is located in the `/usr/lib/locale/`*locale* directory and is called `LC_TIME`. Among other information, this file contains entries for the name of the month (including abbreviations), the names of the weekdays (including abbreviations), and the standard character strings for specifying the local time (`%X`) and date format (`%x`).

LC_TIME Locale Definition

For locale definition, the following mandatory keywords are recognized:

`abmon`
: Define the abbreviated month names, corresponding to the `%b` field descriptor. The operand consists of twelve semicolon-separated strings, each surrounded by double-quotes. The first string is the abbreviated name of the first month of the year (January), the second the abbreviated name of the second month, and so on.

`mon` Define the full month names, corresponding to the `%B` field descriptor. The operand consists of twelve semicolon-separated strings, each surrounded by double-quotes. The first string is the full name of the first month of the year (January), the second the full name of the second month, and so on.

`abday`
: Define the abbreviated weekday names corresponding to the `%a` field descriptor (conversion specification in the `strftime()`, `wcsftime()` and `strptime()` functions). The operand consists of seven semicolon-separated strings, each surrounded by double-quotes. The first string is the abbreviated name of the day corresponding to Sunday, the second the abbreviated name of the day corresponding to Monday, and so on.

`day` Define the full weekday names, corresponding to the `%A` field descriptor. The operand consists of seven

semicolon-separated strings, each surrounded by double-quotes. The first string is the full name of the day corresponding to Sunday, the second the full name of the day corresponding to Monday, and so on.

`t_fmt`
: Define the appropriate time representation, corresponding to the `%X` field descriptor. The operand consists of a string, and can contain any combination of characters and field descriptors. In addition, the string can contain escape sequences.

`d_fmt`
: Define the appropriate date representation, corresponding to the `%x` field descriptor. The operand consists of a string, and can contain any combination of characters and field descriptors. In addition, the string can contain escape sequences.

`d_t_fmt`
: Define the appropriate date and time representation, corresponding to the `%c` field descriptor. The operand consists of a string, and can contain any combination of characters and field descriptors. In addition, the string can contain escape sequences (\\, \a, \b, \f, \n, \r, \t, \v).

`am_pm`
: Define the appropriate representation of the ante meridian and post meridian strings, corresponding to the `%p` field descriptor. The operand consists of two strings, separated by a semicolon, each surrounded by double-quotes. The first string represents the ante meridian designation, the last string the post meridian designation.

`t_fmt_ampm`
: Define the appropriate time representation in the 12-hour clock format with `am_pm`, corresponding to the `%r` field descriptor. The operand consists of a string and can contain any combination of characters and field descriptors. If the string is empty, the 12-hour format is not supported in the locale.

`era`
: Define how years are counted and displayed for each era in a locale. The operand consists of semicolon-separated strings. Each string is an era description segment with the format:

  direction:*offset*:*start_date*:*end_date*:*era_name*:*era_format*

  according to the definitions below. There can be as many era description segments as are necessary to describe the different eras.

  Note: The start of an era might not be the earliest point in the era it may be the latest. For example, the Christian era BC starts on the day before January 1, AD 1, and increases with earlier time.

  direction
  : Either a + or a – character. The + character indicates that years closer to the *start_date* have lower numbers than those closer to the *end_date*. The – character indicates that years closer to the *start_date* have higher numbers than those closer to the *end_date*.

  offset
  : The number of the year closest to the *start_date* in the era, corresponding the `%Ey` field descriptor.

  start_date
  : A date in the form *yyy*/*mm*/*dd*, where *yyyy*, *mm*, and *dd* are the year, month and day numbers respectively of the start of the era. Years prior to AD 1 are represented as negative numbers.

  end_date
  : The ending date of the era, in the same format as the *start_date*, or one of the two special values –* or +*. The value –* indicates that the ending date is the beginning of time. The value +* indicates that the ending date is the end of time.

  era_name
  : A string representing the name of the era, corresponding to the `%EC` field descriptor.

  era_format
  : A string for formatting the year in the era, corresponding to the `%EY` field descriptor.

`era_d_fmt`

Define the format of the date in alternative era notation, corresponding to the `%Ex` field descriptor.

`era_t_fmt`
Define the locale's appropriate alternative time format, corresponding to the `%EX` field descriptor.

`era_d_t_fmt`
Define the locale's appropriate alternative date and time format, corresponding to the `%Ec` field descriptor.

`alt_digits`
Define alternative symbols for digits, corresponding to the `%O` field descriptor modifier. The operand consists of semicolon-separated strings, each surrounded by double-quotes. The first string is the alternative symbol corresponding with zero, the second string the symbol corresponding with one and so on. Up to 100 alternative symbol strings can be specified. The `%O` modifier indicates that the string corresponding to the value specified via the field descriptor will be used instead of the value.

`copy` Specify the name of an existing locale to be used as the definition of this category. If this keyword is specified, no other keyword can be specified.

Each string is on a line by itself. All white space is significant. The order of the strings in the above list is the same as the order in which they must appear in the file.

LC_TIME C-language Access

The following information can be accessed. These correspond to constants defined in `<langinfo.h>` and used as arguments to the `nl_langinfo()` function.

`ABDAY_`*x*
The abbreviated weekday names (for example Sun), where *x* is a number from 1-7.

`DAY_`*x*
The full weekday names (for example Sunday), where *x* is a number from 1 to 7.

`ABMON_`*x*
The abbreviated month names (for example Jan), where *x* is a number from 1 to 12.

`MON_`*x*
The full month names (for example January), where *x* is a number from 1 to 12.

`D_T_FMT`
The appropriate date and time representation.

`D_FMT`
The appropriate date representation.

`T_FMT`
The appropriate time representation.

`AM_STR`
The appropriate ante-meridian affix.

`PM_STR`
The appropriate post-meridian affix.

`T_FMT_AMPM`
The appropriate time representation in the 12-hour clock format with `AM_STR` and `PM_STR`.

`ERA` See description of `era` in "**LC_TIME Locale Definition**" above.

`ERA_D_FMT`
The era date format.

`ERA_T_FMT`
The locale's appropriate alternative time format, corresponding to the `%EX` field descriptor.

`ERA_D_T_FMT`
The locale's appropriate alternative date and time format, corresponding to the `%Ec` field descriptor.

`ALT_DIGITS`
The alternative symbols for digits, corresponding to the `%O` conversion specification modifier. The value consists of semicolon-separated symbols. The first is the alternative symbol corresponding to zero, the

second is the symbol corresponding to one, and so on. Up to 100 alternative symbols may be specified.

The following table displays the correspondence between the items described above and the conversion specifiers used by the date utility and the `strftime()`, `wcsftime()` and `strptime()` functions.

Localedef Keyword	langinfo Constant	Conversion Specifier
abmon	ABMON_*x*	%b
mon	MON	%B
abday	ABDAY_*x*	%a
day	DAY_*x*	%A
t_fmt	T_FMT	%X
d_fmt	D_FMT	%x
d_t_fmt	D_T_FMT	%c
am_pm	AM_STR	%p
am_pm	PM_STR	%p
t_fmt_ampm	T_FMT_AMPM	%r
era	ERA	%EC, %Ey, %EY
era_d_fmt	ERA_D_FMT	%Ex
era_t_fmt	ERA_T_FMT	%EX
era_d_t_fmt	ERA_D_T_FMT	%Ec
alt_digits	ALT_DIGITS	%O

Each string is on a line by itself. All white space is significant. The order of the strings in the above list is the same as the order in which they must appear in the file.

EXAMPLES

Beginning of `/usr/lib/locale/En_US.ASCII/LC_TIME` file:

```
Jan
Feb
...
January
February
 ...
Sun
Mon
 ...
Sunday
Monday
 ...
%T %Z
%m/%d/%y
%a %h %d %T %Z %Y
AM
PM
 ...
```

SEE ALSO

ctime(3C), nl_langinfo(3C), setlocale(3C), strftime(3C), strptime(3C), wcsftime(3C),

`langinfo(5)` .

System Administrator's Reference Manual

log3admin(4)

NAME

log3admin - logging administration file

SYNOPSIS

$LOGPATH/log3admin

DESCRIPTION

The log server is controlled by means of the log3admin administration file. The administrator uses entries in this file to determine, for example, which log files are to record the log information and whether and to where it is to be forwarded.

If log data for a particular application is to be written to a log file other than that originally planned, it is not necessary to modify the application. In this case, it is sufficient to adjust the relevant entry in the administration file.

The administration file is located in the directory defined by $ADMPATH. If $ADMPATH is not specified in the environment, it is replaced by the value of $LOGPATH

The administration file is an ASCII text file that can also be edited directly using an editor.

The log server does not interpret the log3admin text file directly, rather it expects a log3admin.obj binary file, which it can load into its own data structures and process. This log3admin.obj file is created using the log3adm(1M) command. The log3adm(1M) command must be invoked after every change to the log3admin file, so that the changes are effective. The log3adm command carries out a syntax test and creates the log3admin.obj binary file if there are no errors.

If the administration file is modified while the log server is running, the log server can be instructed to update its data using the SIGUSER signal.

The log3admin administration file contains four information blocks, all of which, except for the PROTOCOL block, must be specified. These blocks can be written to the file in any order.

Every block begins with one of the following keywords

#LOGFILE
: Properties of the logbooks

#LOGTYPE
: Assignment of the applications to the logbooks, etc.

#DEFAULT
: Description of the default logbook

#PROTOCOL
: Description of the ASCII protocol file (also contains individual files)

and ends with the first # character that does not occur in a comment field. All characters that come after a semicolon (;) to the end of the line, as well as the lines between blocks, are interpreted as comment. Both blank lines and pure comment lines are permitted.

Description of the #LOGFILE block

Each administration entry in the #LOGFILE block comprises a line with six fields, which are separated from one another by blanks or tabs:

FILENAME SIZE CYCLE CLOSE COMMAND COMMENT

These fields are described below.

FILENAME
: This is the name of the logbook. It can only contain the characters [A to Z], [a to z], [0 to 9], underscore (_), slash (/) and period (.).

  Slashes (/) can only be used if the name contains an absolute path. If there is no absolute path, the

logbook files are created relative to $LOGPATH.

The length of a name with an absolute path is restricted to 32 characters. The length of a simple name must not exceed ten characters because the log server appends a 4-character suffix to it, and some systems only differentiate file names up to the 14th character.

Several applications (components) can write to a logbook.

SIZE This specifies the maximum permitted size of the individual log files in a logbook in Kbytes.

The lower limit is defined by means of the size LOGSIZE (plus one Kbyte) defined in the include file logging.h, so that at least one log entry can be written to a file.

CYCLE

The number of files in a logbook is entered in this field. Digits between 2 and 9 are permitted.

The disk storage capacity required can be estimated using the formula:

Requirement=Sum of SIZE[i]*CYCLE[i]

(i = Number of log files)

CLOSE

This tells the log server either to close the currently open log file after a logging procedure or to keep it open. Permitted entries are 1 ("close") and 0 ("open"). In order to use as few system resources as possible (number of files open simultaneously), this flag should be set to 1 ("close") for files which are written to rarely.

COMMAND

This field is provided in order to allow an instruction sequence to be specified optionally in inverted commas ("..."), which is then executed with root authorization.

When a log file has reached its maximum size, an instruction entered here is carried out by the log server using the functions fork() and execlp(), where the name of the log file is transferred as a parameter. The log server does not wait for this process to end.

The instruction must either contain an absolute path or be integrated in the $PATH environment of the log server.

If an instruction sequence itself contains inverted commas, they must follow a backslash (\). A complete log file can be archived in this way.

An instruction can be a command (e.g. rm), a shell script or an executable C program. If a shell script is used, it should begin with the line #!/bin/sh. A message for executing this instruction and an error message are entered in the $LOGPATH/debug_log3 debug file.

COMMENT

Any comment.

Description of the #LOGTYPE block

Every entry in the #LOGTYPE block comprises one line with five fields separated by means of blanks or tabs:

COMP NO LOGFILE REPEATS DIRECTION COMP NAME

COMP NO

Component number: This field is intended for the unique component number assigned to every Logging V3.0 application by operating system development at FSC.

A * character instead of a component number means that all applications can write to this logbook.

LOGFILE

The component number is assigned to a logbook. This logbook must be defined in the #LOGFILE block.

A component may write to several logbooks.

A * character instead of a logbook name means that this application writes to all logbooks defined in the #LOGFILE block.

REPEATS

Permitted entries are 1 (repeat recognition required) and 0 (no repeat recognition).

If repeat recognition is required for this component, the log server always compares the last log entry with the current one before it writes to this logbook. If the two entries are the same, the repetition counter and the repetition time are simply updated in the last log entry.

This entry can be overridden when the log server is started.

DIRECTION

Permitted entries are:

- 0 = logging only
- 1 = pass the log data on to TMS (Threshold Monitoring System)
- 2 = filter the log data and pass it to Event Management, so that any necessary reactions are initiated
- 3 = 1 + 2
- 4 = store log data in ASCII files (for forwarding to administration tools such as TRANSVIEW or Unicenter TNG)
- 5 = 1 + 4
- 6 = 2 + 4
- 7 = 3 + 4

The log data of components, which are not defined in the #LOGTYPE block, is written to the default logbook. In this case, the log entries are not forwarded but the repeat recognition remains active.

The following special default applies for forwarding data to Event Management: All log entries, which are not assigned explicitly to a logbook, are passed to Event Management with a component number. The forwarding can only be canceled by explicitly assigning the respective component number to a logbook.

Forwarding to TMS and ASCII logging can, of course, only be executed if the log server was started with the options for starting the necessary daemons and if the daemons could also be started properly.

COMP NAME

A meaningful name for the component can be specified here. The name is treated as comment and only used for selecting a component number.

The syntax of the #DEFAULT block is identical to that of the #LOGFILE block. The only difference is that this block only contains one entry, which is mandatory (see sample file).

The syntax of the #PROTOCOL block is identical to that of the #LOGFILE block. The only difference is that only one entry is permitted here for the ASCII protocol file. Furthermore, only the size of the files can be modified.

EXAMPLES

```
#DEFAULT
;FILENAME       SIZE      CYCLE CLOSE          COMMAND      COMMENT
;-------------  --------- ----- ------------- ----------   -----------------
;file 10 chars  in Kbyte  2...9 0: keep open  "string"
;path 32 chars  min. 3          1: close file
default         400       9     0                        ; default logbook
for components without #LOGTYPE declaration (must not be renamed or deleted)
#PROTOCOL
;FILENAME       SIZE      CYCLE CLOSE          COMMAND      COMMENT
;-------------  --------- ----- ------------- ----------   -----------------
;fix            in Kbyte  fix   fix           no
;               min. 3
protocol        1000      2     1                        ; ASCII protocol
file (only size can be changed)
#LOGFILE
;FILENAME       SIZE      CYCLE CLOSE          COMMAND      COMMENT
;-------------  --------- ----- ------------- ----------   -----------------
```

```
;file 10 chars  in Kbyte  2...9 0: keep open  "string"
;path 32 chars  min. 3          1: close file
datasyslog      250       4     0                          ; system logbook
#LOGTYPE
;COMP NO    LOGFILE    REPEATS  DIRECTION                    COMP NAME
;---------- ---------- -------- ------------------------   ----------------
;positive   defined    1:yes    0:Logging V3.0 only
;component  logfile    0:ignore 1:Threshold Monitoring System (TMS)
;number                         2:Event management
;                               4:ASCII protocol
;                               combination possible
;                               e.g. 5=TMS+ASCIIprotocol
1           datasyslog 1        6                          ; Logging V3.0
4294967270  default    1        6                          ; local7
4294967271  default    1        6                          ; local6
4294967272  default    1        6                          ; local5
4294967273  default    1        6                          ; local4
4294967274  default    1        6                          ; local3
4294967275  default    1        6                          ; local2
4294967276  default    1        6                          ; local1
4294967277  default    1        6                          ; local0
4294967278  datasyslog 1        6                          ; cron
4294967279  datasyslog 1        6                          ; uucp
4294967280  datasyslog 1        6                          ; news
4294967281  datasyslog 1        6                          ; lpr
4294967282  datasyslog 1        6                          ; auth
4294967283  datasyslog 1        6                          ; daemon
4294967284  datasyslog 1        6                          ; mail
4294967285  datasyslog 1        6                          ; user
4294967286  datasyslog 1        6                          ; syslog
4294967290  log3debug  1        0                          ; Mark
4294967292  datasyslog 1        7                          ; driver/cmn/SNS
4294967293  datasyslog 1        6                          ; kernel
4294967294  datasyslog 1        6                          ; syslogd
1080001     datasyslog 1        6                          ; Config
1080002     datasyslog 1        6                          ; USV
1080006     default    1        6                          ; hl_muci
1080007     default    1        6                          ; Networker
1080008     default    1        6                          ; cleandev
1080009     datasyslog 1        6                          ; Diagnostic LAR
1080010     datasyslog 1        6                          ; errlogd
1080011     default    1        6                          ; xbload
1080012     default    1        6                          ; ATLS
1080013     default    1        6                          ; write cache
1080014     default    1        6                          ; SPOOL V4.2B
1080015     default    1        6                          ; RAID PXRE/C
1080016     default    1        6                          ; chat-logouts
1080017     default    1        6                          ; chat-periphery
1080018     default    1        6                          ; chat-texts
1080019     default    1        6                          ; Cluster OS
1080020     default    1        6                          ; Informix
1080021     default    1        6                          ; Oracle
1080022     default    1        6                          ; Cross-Basic
1080023     default    1        6                          ; RMS
1100000     default    1        6                          ; SNMP Agent
1130010     default    1        6                          ; ASX
#end
```

FILES

`$LOGPATH/log3admin.obj`

SEE ALSO

`log3adm`(1M), `log3str`(1M), `syslogd`(1M), `log3struct`(4), `log3`(5).

System Administrator's Reference Manual

log3status(4)

NAME

`log3status` - runtime status of Logging V3.0

SYNOPSIS

`$LOGPATH/log3status`

DESCRIPTION

The `log3status` file is always created by the `log3svr` log server under `$LOGPATH` and read from there both by the `log3get(1M) GET` command and by the `/etc/init.d/log3` start script (or `/etc/rc2.d/S06log3`).

The defined attributes essentially describe the call parameters for the log server, the paths of the communication pipes opened by the log server and the PIDs of the services started and monitored by the log server.

Attribute definitions:

`alarm interval`
: Minutes between two MARK entries if no log messages are received for a logbook (default: 1440 minutes = 1 day)

`repeatings`
: Enable counting of log entries with the same message content (default: yes)

`debug mode`
: Enable debug mode (default: no)

`event management`
: Start Event Management (default: yes)

`all to TransView`
: Write readable ASCII protocol file for TransView (default: no)

`event reporting`
: Start DSM-ER Event Reporting (default: no)

`support T-Log2.0`
: Start T-Log 2.0 converter (default: no)

`log3 pipe`
: `/opt/log3/log3pipe`

`event pipe`
: `/opt/log3/analyze_pipe`

`protocol pipe`
: `/opt/log3/protocol_pipe`

`report pipe`
: `/er/ERDIAGPIPE`

`log3svr pid`
: PID of `log3svr` log server

`log3tlog2 pid`
: PID of the `log3tlog2` T-Log 2.0 converter

`log3analyze pid`
: PID of the `log3analyze` Event Management daemon

`log3trans pid`
: PID of the `log3trans` protocol daemon

Each line in `$LOGPATH/log3status` comprises two fields separated by "=":

Field1: Attribute name

2:	Current value at runtime

EXAMPLES

Contents of the `log3status` file:

```
alarm interval=1440
repeatings=yes
debug mode=no
event management=yes
all to TransView=no
event reporting=no
support T-Log2.0=no
log3 pipe=/opt/log3/log3pipe
event pipe=/opt/log3/analyze_pipe
protocol pipe=/opt/log3/protocol_pipe
report pipe=/er/ERDIAGPIPE
log3svr pid=5440
log3tlog2 pid=-1
log3analyze pid=5441
log3trans pid=-1
```

NOTES

The file is fully overwritten by the `log3svr` log server at startup, as well as every time it registers a change during operation (e.g. termination of a service) and when it is stopped. External changes to the file are thus lost during operation. However, when Logging V3.0 is restarted, the call parameters for the log server can be changed between the time the log server is stopped and restarted. When Logging V3.0 is started via the `/etc/init.s/log3` start script, an attempt is made to take the call parameters from `log3status`. This approach guarantees that any call parameters stored by the log server in `log3status` are used again the next time it is started. Logging V3.0 must be stopped in order to change the call parameters in the file. These changes then become effective for a subsequent startup.

If the call parameters are to be changed during operation (i.e. Logging V3.0 is to run directly with the modified call parameters), they can be transferred as an option to the `/etc/init.d/log3` start script (example: `/etc/init.d/log3 start –d 3 –m 20`). `log3status` is not evaluated in this case, rather the log server is stopped and started with the transferred call parameters. The log server itself then saves the new call parameters in `log3status` for further restarts.

While the names of the communication pipes are predefined, their paths can be changed by environment variables.

The PIDs are pure runtime sizes and change with each restart.

FILES

`$LOGPATH/debug_log3`

SEE ALSO

`log3get`(1M), `log3`(5).

System Administrator's Reference Manual

log3struct(4)

NAME

`log3struct` - Logging description file

SYNOPSIS

`$LOGPATH/log3struct`

DESCRIPTION

The administrator defines the layout descriptions of the log data in the `log3struct` description file which the `log3display(1M)` command requires to display this data. Here, the administrator assigns various display options to the numbers of the descriptions (also called structure IDs). The `log3display(1M)` command interprets these options to edit the application data.

The application transfers the value of the structure ID to the log server as a parameter of the `log3write(5)` function.

The description file is located in the directory defined by `$ADMPATH`. If `$ADMPATH` is not specified in the environment, it is replaced by the value of `$LOGPATH`.

The `log3struct` file is an ASCII text file that can be edited either directly with an editor or via the menu.

The `log3display(1M)` command does not interpret the `log3struct` text file directly, rather it waits for a `log3struct.obj` binary file which it can load and process in its own data structures.

This `log3struct.obj` file is created using the `log3str` command, which is called automatically via the menu. After a description file has been modified directly using the editor, the `log3str` command must be called from the shell level.

The `log3str` command carries out a syntax test and creates the `log3struct.obj` binary file if there are no errors.

Options can be used to suppress or force the display of most fields. The `log3display(1M)` command uses a layout description for the log data blocks, – if a structure ID is available – for example, in order to supply the individual blocks with a meaningful identifier.

A layout description is not required for log data blocks with the structure ID 0, therefore it need not be entered.

If there is no suitable layout description for the structure ID (`STRUCT` statement), every log data block is displayed on a new line (with no identifier).

Syntax of file log3struct

Every description begins with the following ID:

`#STRUCT` *number* ; (*number* = 1 ... 65534)

and ends with the first # character, that is not contained in a comment field.

Comments include all characters that come after a semicolon (;) to the end of the line, and all lines after a # character has been recognized up to the next `#STRUCT` statement. Empty lines and pure comment lines are permitted everywhere.

A number between `1` and `65534` follows the `#STRUCT` statement, representing the structure ID from the log data header.

The following lines contain a string in inverted commas "..." for each data block, and this string can also include special characters, such as \n, \r, \t, \b, \000 to \377.

If the number of strings specified exceeds the number of log data blocks in the log information, the remaining strings are appended to the evaluation. If the opposite is true, the remaining data blocks are displayed as though there were no layout description.

Example of a description file

```
; #STRUCT 172   EXAMPLE
; "\nText1   "
; "~HEXDUMP:\n"  ; Tilde (~) at the 1st position of a string for a
;                ; HEXDUMP produces a display similar to the UNIX
;                ; instruction 'od'. '~' is not displayed!
; ""             ; Nothing should be output before the data field.
; "\nText2  "
#STRUCT 65534    ; syslogd
"\n"
"\nSYSTEM:"
"  DATE:"
;------------ T-LOG 2.0 structure descriptions  --------------
#STRUCT 3      ; T-LOG 2.0
"\n"
#STRUCT 4      ; T-LOG 2.0
"\ntype: "
"\nPGM: "
"\ntine no.: "
"\nerror no.: "
"\nUNIX errno: "
"\ntext: "
#STRUCT 13     ; T-LOG 2.0
"\ntest device: "
"\nerror number: "
"\nfile: "
"\nrelease: "
"\nline: "
"~ot_mess" ; ~<Name of message file> - MCS
```

SEE ALSO

log3display(1M), syslogd(1M), log3admin(4), log3(5).

System Administrator's Reference Manual

login(4)

NAME

`login` - /etc/default/login file

DESCRIPTION

The `/etc/default/login` file contains the default settings for the `login(1)` command.

FORMAT

The format of this file is *identifier=value* where both the identifiers and values are defined by the `login(1)` program.

CONTENTS

`ALTSHELL`
: Set to `YES` if a shell other than the default shell is permitted in `/etc/passwd`.

`CONSOLE`
: `root` may only log in on the path name at which `CONSOLE` is set. Typically, this would be `/dev/console`.

`COVERTLOGIN`
: If set to `YES`, the user login name will not be echoed.

`DCE_LOGIN`
: If `DCE_LOGIN` is set to `YES`, DCE users can log in by accessing the DCE registry, provided the DCE optional product is installed on the system.

`DISABLE_RHOSTS`
: If `DISABLE_RHOSTS` is set to `YES`, then the `.rhosts` file is ignored and all remote logins will be subjected to the same login requirements as local users.

`HZ` The system clock rate. This value should always be set to `100`.

`IDLEWEEKS`
: Sets a predetermined expiry date for user passwords. If a password exists for longer than `IDLEWEEKS`, it becomes invalid and the user will not be able to log in.

`IGNORE_UPPERCASE`
: If `IGNORE_UPPERCASE` is set to `YES`, `login` should not check for "dumb" uppercase terminals. Otherwise, if `IGNORE_UPPERCASE` is set to `NO` and the login name only contains uppercase letters, `login` assumes a "dumb" terminal and changes I/O control so that all input/output is mapped to uppercase letters.

`LOGIN_DELAY`
: Each unsuccessful attempt to login doubles the delay until the next login prompt. The maximum delay is set by `LOGIN_DELAY`.

`MAXAUTH`
: If a user attempts to log in more than `MAXAUTH` times using an invalid password, the user account is locked (see also `passwd -l`). This lock can only be removed by the superuser (creating a new password). The password lock can be removed by setting `MAXAUTH=0` or by commenting out or deleting the entry.

`MAXTRIES`
: This feature will allow the number of guesses before dropping carrier to be tuned.

  `MAXTRIES=0`
  : No limit on the number of guesses.

  `MAXTRIES=>0`
  : Carrier will be dropped after `MAXTRIES` tries.

  `MAXTRIES=<0`

No limit on the number of guesses, but all guesses after abs(MAXTRIES) will fail even if the password is correct. If LOCKTTY is set, hang-ups will be ignored – and the tty will remain locked until the administrator (root) kills the login process (signal –9 or similar).

PATH Sets the default path for the login session of a user who does not have the uid 0.

STRICT_RHOST_MODES
If STRICT_RHOST_MODES is set to YES, then a user's local .rhost file must not be rwx for the group or for other. The file must (always) only be owned by the user or root.

SUPATH
Sets the default path for the root login session.

TIMEOUT
If the login does not succeed within the time set in TIMEOUT seconds, login(1) terminates.

TIMEZONE
The default value for TIMEZONE is defined in /etc/TIMEZONE [see timezone(4)]. To change the default time zone, use /etc/TIMEZONE. Do not set TIMEZONE in /etc/default/login, as this will override the value in /etc/TIMEZONE.

ULIMIT
This variable sets the maximum file size for a user. It is in units of 512-byte blocks.

ULIMIT=UNLIMITED means: The file size is not limited (except where the file size is defined by the respective file system implementation).

If there is no ULIMIT entry or if the entry is commented out, the file size defined in the mtune(4) or stune(4) file is adopted.

UMASK
Sets the default umask for this login session.

The following variables are only used if the CSP (C2 Security) optional product is installed.

AUTO_LOGIN_MIN_UID
This is to set the minimum UID and can use .rhosts to log in without typing a password. Note that if the CSP (C2 Security) optional product is installed, an entry in /etc/hosts.deny overrides anything in .rhosts and /etc/hosts.equiv and results in a prompt for a password. The format of /etc/hosts.deny is identical to /etc/hosts.equiv.

CHECK_ROOT_GROUP
If GET_REAL_NAME is also true, this forces the real name to be a member of group root before allowing root to log in.

CHECKGROUP
If not set no, then a non-root user must be a member of their login group when they log in.

GET_REAL_NAME
When logging in as root, this forces a real login name to be supplied.

GRACE
GRACE seconds before the login session is terminated, or a warning message is printed on the controlling tty. If GRACE is negative, the session is not forced to end.

LOCKTTY
If LOCKTTY is not no then a port with too many failed guesses is disabled and the system administrator must reenable it. MAXTRIES must be negative for this to be enabled.

MAXUNAUTH
If a user tries to log in more than MAXUNAUTH times during an unauthorized time, the user is locked. This only applies if timed login sessions are enabled.

SECUREUSER
The login name of the system security officer, root, and this login account are always allowed to try to log in.

TIME_WINDOWS
If this is not set to no, then timed login sessions are enabled.

DEFAULT VALUES

```
ALTSHELL=YES
AUTO_LOGIN_MIN_UID=0
CHECK_ROOT_GROUP=NO
CHECKGROUP=NO
CONSOLE=/dev/console
COVERTLOGIN=YES
DCE_LOGIN=NO
DISABLE_RHOSTS=YES
GET_REAL_NAME=NO
GRACE=60
HZ=100
IDLEWEEKS=-1
IGNORE_UPPERCASE=YES
LOCKTTY=NO
LOGIN_DELAY=61
MAXAUTH=5
MAXTRIES=5
MAXUNAUTH=2
PATH=/usr/bin:/usr/sbin:/opt/bin
SECUREUSER=sso
STRICT_RHOST_MODES=NO
SUPATH=/sbin:/usr/sbin:/usr/bin:/opt/bin
TIME_WINDOWS=NO
TIMEOUT=60
TIMEZONE=EST5EDT
ULIMIT=32768
UMASK=022
```

SEE ALSO

login(1).

System Administrator's Reference Manual

loginlog(4)

NAME

`loginlog -` log of failed login attempts

SYNOPSIS

`/var/adm/loginlog`

DESCRIPTION

After five unsuccessful login attempts, all the previous attempts are logged in the `/var/adm/loginlog` file. This file contains one record for each failed attempt. Each record contains the login name, tty specification, and time.

This is an ASCII file. Each field within each entry is separated from the next by a colon. Each entry is separated from the next by a new line.

By default, `loginlog` does not exist, so no logging is done. To enable logging, the log file must be created with read and write permission for owner only. The owner must be `root` and group must be `sys`.

SEE ALSO

`login(1)`, `passwd(1)`.

System Administrator's Reference Manual

mailcnfg(4)

NAME

`mailcnfg` - initialization information for mail and rmail

SYNOPSIS

`/etc/mail/mailcnfg`

DESCRIPTION

The `/etc/mail/mailcnfg` file contains initialization information for the `mail` and `rmail` commands. Each entry in `mailcnfg` consists of a line of the form

Keyword = *Value*

Leading whitespace, whitespace surrounding the equal sign, and trailing whitespace is ignored. *Keyword* may not contain embedded whitespace, but whitespace may appear within *Value*. Undefined keywords or badly formed entries are silently ignored.

Keyword Definitions

`DEBUG`

Takes the same values as the `-x` invocation option of `mail`. This provides a way of setting a system-wide debug/tracing level. Typically `DEBUG` is set to a value of 2, which provides minimal diagnostics useful for debugging `mail` and `rmail` failures. The value of the `-x mail` invocation option will override any specification of `DEBUG` in `mailcnfg`.

`CLUSTER`

To identify a closely coupled set of systems by one name to all other systems, set *Value* to the cluster name. This string is used to supply the `...remote from...` information on the `From` header line rather than the system nodename returned by `uname(2)` .

`FAILSAFE`

In the event that the `/var/mail` directory is accessed through NFS within a cluster provisions must be made to allow for the directory not being available when local mail is to be delivered (remote system crash, NFS problems). *Value* is a string that indicates where to forward the current message for delivery. Typically this is the remote system that actually owns `/var/mail`. In this way, the message is queued for delivery to that system when it becomes available. For example, assume a cluster of systems (`sysa`, `sysb`, `sysc`) where `/var/mail` is physically mounted on `sysc` and made available to the other machines through NFS. If `sysc` were to crash, the NFS-accessible `/var/mail` would become unavailable and local deliveries of mail would go to `/var/mail` on the local system. When `/var/mail` is re-mounted through NFS, all messages deposited in the local directory would be hidden and essentially lost. To prevent this, if `FAILSAFE` is defined in `mailcnfg`, `mail` and `rmail` check for the existence of `/var/mail/:saved`, a required subdirectory. If this subdirectory does not exist, `mail` assumes that the NFS-accessible `/var/mail` is not available and invokes the failsafe mechanism of automatically forwarding the message to *Value*. In this example *Value* would be `sysc!%n`. The `%`*n* keyword is expanded to be the recipient name and thus the message would be forwarded to `sysc!`*recipient_name*. Because `sysc` is not available, the message remains on the local system until `sysc` is available, and then sent there for delivery.

`DEL_EMPTY_MFILE`

If not specified, the default action of `mail` and `rmail` is to delete empty mailfiles if the permissions are 0660 and to retain empty mailfiles if the permissions are anything else. If *Value* is `yes`, empty mailfiles are always deleted, regardless of file permissions. If *Value* is `no`, empty mailfiles are never deleted.

`DOMAIN`

This string is used to supply the system domain name in place of the domain name returned by `getdomainname`.

`SMARTERHOST`

This string may be set to a smarter host which may be referenced within the mail surrogate file through `X`.

mailsurr_keyword

Certain predefined single letter keywords are textually substituted in surrogate command fields before they are executed [see `mailsurr(4)`]. While none of the predefined keywords may be changed in meaning, new ones may be defined to provide a shorthand notation for long strings (such as `/usr/lib/mail/surrcmd`) which may appear repeatedly within the `mailsurr` file. Uppercase letters are reserved for future use and will be ignored if encountered here.

NOTES

If `/var/mail` is accessed through NFS and the subdirectory `/var/mail/:saved` is not removed from the local system, the `FAILSAFE` mechanism will be subverted.

FILES

`/etc/mail/mailsurr`
`/var/mail/:saved`
`/usr/lib/mail/surrcmd`

SEE ALSO

`mail(1)`, `uname(2)`, `mailsurr(4)`.

System Administrator's Reference Manual

mailsurr(4)

NAME

`mailsurr` - surrogate commands for routing and transport of mail

SYNOPSIS

`/etc/mail/mailsurr`

DESCRIPTION

The `mailsurr` file contains routing and transport surrogate commands used by the `mail` command. Each entry in `mailsurr` has three whitespace-separated, single quote delimited fields:

'*sender*' '*recipient*' '*command*'

or a line that begins

`Defaults:`

Entries and fields may span multiple lines, but leading whitespace on field continuation lines is ignored. Fields must be less than 1024 characters long after expansion.

The *sender* and *recipient* fields are regular expressions. If the *sender* and *recipient* fields match those of the message currently being processed, the associated command is invoked.

The *command* field may have one of the following five forms:

`A`[`ccept`]
`D`[`eny`]
`T`[`ranslate`] `R=`[|]*string*
`<` `S=...;C=...;F=...;` *command*
`>` *command*

Regular Expressions

The *sender* and *recipient* fields are composed of regular expressions (REs) which are digested by the `regexp`(5) `compile` and `advance` procedures in the C library. The regular expressions matched are those from `ed`(1), with simple parentheses `()` playing the role of `\(\)` and the addition of the `+` and `?` operators from `egrep`(1). Any single quotes embedded within the REs must be escaped by prepending them with a backslash or the RE is not interpreted properly.

The `mail` command prepends a circumflex (`^`) to the start and appends a dollar sign (`$`) to the end of each RE so that it matches the entire string. Therefore it would be an error to use `^`*RE*`$` in the *sender* and *recipient* fields. To provide case insensitivity, all REs are converted to lowercase before compilation, and all sender and recipient information is converted to lowercase before comparison. This conversion is done only for the purposes of RE pattern matching; the information contained within the message's header is not modified.

The sub-expression pattern matching capabilities of `regexp` may be used in the command field, that is, `(...)`, where $1 \le n \le 9$. Any occurrences of `\\`*n* in the replacement string are themselves replaced by the corresponding `(...)` substring in the matched pattern. The sub-expression fields from both the *sender* and *recipient* fields are accessible, with the fields numbered 1 to 9 from left to right.

Accept and Deny Commands

`Accept` instructs `rmail` to continue its processing with the `mailsurr` file, but to ignore any subsequent matching `Deny`. That is, unconditionally accept this message for delivery processing. `Deny` instructs `rmail` to stop processing the `mailsurr` file and to send a negative delivery notification to the originator of the message. Whichever is encountered first takes precedence.

Translate Command

`Translate` allows optional on-the-fly translation of recipient address information. The recipient replacement string is specified as `R=`*string*.

For example, given a command line of the form

```
'.+' '([^!]+)@(.+)\.EUO\.ATT\.com' 'Translate R=attmail!\\2!\\1'
```

and a recipient address of `rob@sysa.EUO.ATT.COM` the resulting recipient address would be `attmail!sysa!rob`.

Should the first character after the equal sign be a "|", the remainder of the string is taken as a command line to be directly executed by `rmail`. If any `sh(1)` syntax is required (metacharacters or redirection), then the surrogate command must be of the form:

`sh -c "`*shell command line ...*`"`

Special care must be taken to escape properly any embedded back-slashes and single or double quotes, since `rmail` uses double quoting to group whitespace delimited fields that are meant to be considered as a single argument to `execl(2)`. It is assumed that the executed command will write one or more replacement strings on `stdout`, one per line. If more than one line is returned, each is assumed to be a different recipient for the message. This mechanism is useful for mailing list expansions. As stated above, any occurrences of `\\`*n* are replaced by the appropriate substring before the command is executed. If the invoked command does not return at least one replacement string (no output or just a newline), the original string is not modified. For example, the command line

```
'.+'        '(.+)' 'Translate R=|/usr/bin/findpath \\1'
```

allows local routing decisions to be made.

If the recipient address string is modified, `mailsurr` is rescanned from the beginning with the new address, and any prior determination of `Accept` is discarded.

< command

The intent of a `<` command is that it is invoked as part of the transport and delivery mechanism, with the ready-for-delivery message available to the command at its standard input. As such, there are three conditions possible when the command exits:

Success
: The command successfully delivered the message. What actually constitutes successful delivery may be different within the context of different surrogates. The `rmail` process assumes that no more processing is required for the message for the current recipient.

Continue
: The command performed some function (logging remote message traffic, for example) but did not do what would be considered message delivery. The `rmail` process continues to scan the `mailsurr` file looking for some other delivery mechanism.

Failure
: The command encountered some catastrophic failure. The `rmail` process stops processing the message and sends to the originator of the message a non-delivery notification that includes any `stdout` and `stderr` output generated by the command.

The semantics of the `<` command field in the `mailsurr` file allow the specification of exit codes that constitute success, continue, and failure for each surrogate command individually. The syntax of the exit state specification is:

`<` WS [*exit_state_id*=*ec*[`,` *ec*[,...]]`;`][*exit_state_id*=*ec*[,*ec*[,...]]`;` [...]]] WS*surrogate_cmd_line*

WS is whitespace. *exit_state_id* can have the value `S`, `C`, or `F`. *exit_state_id*s can be specified in any order. *ec* can be:

- any integer $0 \le n \le 255$ [Negative exit values are not possible. See `exit(2)` and `wait(2)`.]
- arange of integers of the form *lower_limit–upper_limit* where the limits are ≥ 0 and ≤ 255, and
- `*`, which implies *anything*

For example, a *command* field of the form:

'`< S=1-5,99;C=0,12;F=*;` *command* %R'

indicates that exit values of 1 through 5, and 99, are to be considered success, values of 0 (zero) and 12 indicate continue, and that anything else implies failure. If not explicitly supplied, default settings are `S=0;C=*;`.

It may be possible for ambiguous entries to exist if two exit states have the same value, for example, `S=12,23;C=*;F=23,52`; or `S=*;C=9;F=*;`. To account for this, `rmail` looks for explicit exit values (that is, not *) in order of success, continue, failure. Not finding an explicit match, `rmail` then scans for * in the same order.

It is possible to eliminate an exit state completely by setting that state's value to an impossible number. Since exit values must be between 0 and 255 (inclusive), a value of 256 is a good one to use. For example, if you had a surrogate command that was to log all message traffic, a `mailsurr` entry of

`'(.+)''(.+)''<`S=256;C=*;/usr/lib/mail/surrcmd/logger \\1 \\2'

would always indicate continue. Surrogate commands are executed by `rmail` directly. If any shell syntax is required (metacharacters, redirection, etc.), then the surrogate command must be of the form:

`sh –c "`*shell command line ...*`"`

Special care must be taken to properly escape any embedded backslashes and other characters special to the shell.

If there are no matching `<` commands, or all matching `<` commands exit with a continue indication, `rmail` attempts to deliver the message itself by assuming that the recipient is local and delivering the message to `/var/mail/recipient`.

> command

The intent of a `>` command is that it is invoked after a successful delivery to do any post-delivery processing that may be required. Matching `>` commands are only executed if a `<` command indicates a successful delivery (see the previous section) or local delivery processing is successful. The `mailsurr` file is rescanned and all matching `>` commands, not just those following the successful `<` command, are executed in order. The exit status of an `>` command is ignored.

Defaults: Line

The default settings may be redefined by creating a separate line in the `mailsurr` file of the form

`Defaults: [S=...;][C=...;][F=...;]`

`Defaults:` lines are honored and the indicated default values redefined when the line is encountered during the normal processing of the `mailsurr` file. Therefore, to redefine the defaults globally, the `Defaults:` line should be the first line in the file. It is possible to have multiple `Defaults:` lines in the `mailsurr` file, where each subsequent line overrides the previous one.

Surrogate Command Keyword Replacement

Certain special sequences are textually-substituted in surrogate commands before they are invoked:

- `%n` Recipient's full name.
- `%R` Full return path to the originator (useful for sending replies or delivery failure notifications.)
- `%c` Value of the `Content-Type:` header line if present.
- `%C` `text` or `binary`, depending on an actual scan of the content. This is independent of the value of any `Content-Type` header line encountered (useful when calling `ckbinarsys`.)
- `%S` Value of the `Subject:` header line, if present.
- `%l` Value of the `Content-Length:` header line.
- `%L` Local system name. This will be either `CLUSTER` from `mailcnfg` or the value returned by `uname`.
- `%U` Local system name, as returned by `uname`.
- `%X` Value of `SMARTERHOST` in `mailcnfg`.
- `%D` Local domain name. This will be either `DOMAIN` from `mailcnfg`, or the value returned by

`getdomainname`.

`\\`*n* Corresponding (...) substring in the matched patterns. This implies that the `regexp` limitation of 9 substrings is applied to the sender and recipient REs collectively.

`%`*keywords*
Other keywords as specified in `/etc/mail/mailcnfg`. See `mailcnfg(4)`.

The sequences `%L`, `%U`, `%D`, and `%`*keywords* are permitted within the sender and recipient fields as well as in the command fields.

An example of the `mailsurr` entry that replaces the `uux` built-in of previous versions of `rmail` looks like this:

```
'.+'        '([^@!]+)!(.+)' '< /usr/bin/uux - \\1!rmail (\\2)'
```

Mail Surrogate Examples

Some examples of mail surrogates include the distribution of message-waiting notifications to LAN-based recipients and lighting message-waiting lamps, the ability to mail output to printers, and the logging of all `rmail` requests between remote systems (messages passing through the local system). The following is a sample `mailsurr` file:

```
# Some common remote mail surrogates follow. To activate any
# or all of them, remove the # (comment indicators) from
# the beginning of the appropriate lines. Remember that they
# will be tried in the order they are encountered in the file,
# so put preferred surrogates first.
# Prevent all shell meta-characters
ERROR.+ERROR  ERROR.*[`;&|^<>()].*ERROR       ERRORDenyERROR
# Map all names of the form local-machine!user -> user
ERROR.+ERROR  ERROR%L!(.+)ERROR               ERRORTranslate R=\1ERROR
# Map all names of the form uname!user -> user
# Must be turned on when using mail in a cluster environment.
ERROR.+ERROR   ERROR%U!(.+)ERROR               ERRORTranslate R=\1ERROR
# Map all names of the form user@host -> host!user
ERROR.+ERROR  ERROR([^!@]+)@(.+)ERROR         ERRORTranslate R=\2!\1ERROR
# Map all names of the form host.uucp!user -> host!user
ERROR.+ERROR  ERROR([^!@]+)\.uucp!(.+)ERROR  ERRORTranslate R=\1!\2ERROR
# Map all names of the form host.local-domain!user -> host!user
#       DOMAIN= within /etc/mail/mailcnfg will override getdomainname.
ERROR.+ERROR  ERROR([^!@]+)%D!(.+)ERROR       ERRORTranslate R=\1!\2ERROR
#       Allow access to ERRORattmailERROR from remote system ERRORsysaERROR
ERRORsysa!.*ERROR     ERRORattmail!.+ERROR    ERRORAcceptERROR
#       Deny access to ERRORattmailERROR from all other remotes
ERROR.+!.+ERROR       ERRORattmail!.+ERROR    ERRORDenyERROR
#       Send mail for ERRORlaserERROR to attached laser printer
# Make certain that failures are reported via return mail.
ERROR.+ERROR  ERRORlaserERROR ERROR< S=0;F=*; lp -dlaserERROR

# Run all local names through the mail alias processor
#
ERROR.+ERROR  ERROR[^!@]+ERROR         ERRORTranslate R=|/usr/bin/mailalias %nERR
OR
# For remote mail via nusend
#ERROR.+ERROR   ERROR([^!]+)!(.+)ERROR    ERROR< /usr/bin/nusend -d \\1 -s -e -!"rmail \\2" -ER
ROR

# For remote mail via usend
ERROR.+ERROR  ERROR([^!]+)!(.+)ERROR    ERROR< /usr/bin/usend -s -d\\1 -uNoLogin -!"rmail \\2" -
 ERROR
```

```
# For remote mail via uucp
ERROR.+ERROR  ERROR([^!@]+)!.+ERROR   ERROR<S=256;C=0;
 /usr/lib/mail/surrcmd/ckbinarsys –t %C –s \\1ERROR
ERROR.+ERROR  ERROR([^!@]+)!(.+)ERROR   ERROR< /usr/bin/uux – \\1!rmail (\\2)ERROR

# For remote mail via smtp
#ERROR.+ERROR   ERROR([^!@]+)!(.+)ERROR         ERROR< /usr/lib/mail/surrcmd/smtp
qer %R %nERROR
# If none of the above work, then let a router change the address.
#ERROR.+ERROR   ERROR.*[!@].*ERROR      ERRORTranslate R=| /usr/lib/mail/surrcmd/
smail -A %nERROR
# If none of the above work, then ship remote mail off to a smarter host.
# Make certain that SMARTERHOST= is defined within /etc/mail/mailcnfg.
#ERROR.+ERROR   ERROR.*[!@].*ERROR              ERRORTranslate R=%X!%nERROR
# Log successful message deliveries
ERROR(.+)ERROR ERROR(.+)ERROR ERROR>/usr/lib/mail/surrcmd/logger \1 \2ERROR
```

Note that invoking `mail` to read mail does not involve the `mailsurr` file or any surrogate processing.

Security

Surrogate commands execute with the permissions of `rmail` (user ID of the invoker, group ID of mail). This allows surrogate commands to validate themselves, checking that their effective group ID was `mail` at invocation time. This requires that all additions to `mailsurr` be scrutinized before insertion to prevent any unauthorized access to users' mail files. All surrogate commands are executed with the path `/usr/lib/mail/surrcmd:/usr/bin`.

Debugging New mailsurr Entries

To debug `mailsurr` files, use the `–T` option of the `mail` command. The `–T` option requires an argument that is taken as the pathname of a test `mailsurr` file. If null (as in `–T ""`), the system `mailsurr` file is used. Enter

`mail –T` *test_filerecipient*

and some trivial message (such as `testing`), followed by a line with either just a dot (`.`) or a cntl-D. The result of using the `–T` option is displayed on standard output and shows the inputs and resulting transformations as `mailsurr` is processed by the `mail` command for the indicated *recipient*.

Mail messages will never be sent or delivered when using the `–T` option.

NOTES

It would be unwise to install new entries into the system `mailsurr` file without verifying at least their syntactic correctness through `mail –T ...`

FILES

`/usr/lib/mail/surrcmd/*`
: Surrogate commands

`/etc/mail/mailcnfg`
: Initialization information for `mail`

SEE ALSO

ed(1), egrep(1), mail(1), sh(1), uux(1), ckbinarsys(1M), exec(2), exit(2), wait(2), popen(3S), mailcnfg(4), regexp(5).

System Administrator's Reference Manual

mapchan(4)

NAME

`mapchan` - format of terminal mapping files

DESCRIPTION

The `mapchan` mechanism supports the mapping of characters input and output on terminals. On the input side, single characters can be mapped to other characters (1-to-1 mapping), and compose and dead key sequences can be defined. On the output side, single characters can be mapped to one or more characters (1-to-N mapping).

A (non-empty) `mapchan` map file is in a number of sections, each section beginning with a line containing the appropriate keyword.

The "input" section describes the mapping rule for input characters. Every non-empty line in this section must consist of a pair of character values. The first value in the line designates the character which is to be mapped to the second character value.

Example:

```
input
     # Mapping rule for input characters
'a'  'b'
     # Character 'a' is mapped to 'b'
```

The "output" section describes the mapping rule for output characters. The first character value in each line is mapped to the character string defined by the rest of the character values in that line.

Example:

```
output
     # Mapping rule for output characters
'a' 'm' 'n' 'o'
     # Character 'a' is mapped to string "mno"
```

The "compose" section of the map file defines compose key sequences. The header line for this section consists of the keyword `compose` and the character defined as the compose key: "`compose` *character*". The lines in this section must contain three character values: the first two form the second and third characters of the compose sequence, the third is the character to which the compose sequence is mapped.

Example:

```
compose 'x'
     # Introduce compose sequences and define compose key 'x'
'a' 'b' 'm'
     # Compose sequence 'x' 'a' 'b' is mapped to character 'm'
```

The "dead" sections of the map file define dead key sequences. Similarly to the "compose" section, the header line for a "dead" section consists of the keyword `dead` and the dead key: "`dead` *character*". The lines in a "dead" section must consist of a pair of character values: the first value is the second character of the dead key sequence, the second is the character to which the dead key sequence is mapped. There must be a separate "dead" section for each dead key.

Example:

```
dead 'y'
     # Define dead key sequences for dead key 'y'
'a' 'm'
     # Dead key sequence 'y' 'a' is mapped to character 'm'
```

A map file may also include a line containing the keyword beep. This keyword causes an audible signal (BEL) to be emitted in the event of invalid input (such as undefined dead or compose key sequences).

The hash character (#) is interpreted as a comment character. All characters between the comment character

and the end of the line will be ignored.

A map file defined as empty consists of a single line containing the keyword `null`. This keyword must not be combined with any other keywords.

A non-empty map file must consist of at least an "input" section and an "output" section, although these sections do not have to contain mapping rules.

Input and output characters which map to themselves do not need to be included in the map file.

The individual characters in a map file can be specified in various ways. The following formats are accepted:

`'b'` acharacter in single quotes

`98` character code in decimal

`0142` character code in octal

`0x62` character code in hexadecimal

`'\142'`
octal character code in single quotes

`'\x62'`
hexadecimal character code in single quotes

The following control characters are also recognized:

`'\n'` newline character

`'\t'` tab

`'\r'` carriage return

`'\b'` backspace

Sample map file:

```
beep  # Audible signal on error
input
     # Mapping rule for input characters
'.' ','
     # Dot mapped to comma
dead '^'
     # Definition of ^ as dead key
' '  '^'
     # ^ followed by blank produces ^
'a' 0342
     # ^ followed by a produces ISO 88591-1 character â
dead 047
     # Definition of ' as dead key
' '  047
     # ' followed by blank produces '
'e' 0351
     # ' followed by e produces ISO 88591-1 character é
compose 037
     # Definition of character ^_ (control underscore) as □
     # compose key
'c' ',' 0347
     # The sequence ^_ c , produces ISO 88591-1 character ç
output
     # Mapping rule for output characters
'&' 'u' 'n' 'd'
     # Character & is mapped to string "and" (1-to-N mapping)
```

The `mapchan` map files by default reside in the `/usr/lib/mapchan` directory. The `mapchan` command performs the following error checks when processing a map file:

- more than one compose key defined
- "input" or "output" section missing
- more than one "input" or "output" section
- 1-to-N mapping of input characters
- input characters mapped to the NIL character
- general syntax errors

NOTES

If an invalid mapping table is set on a terminal, it will generally no longer be possible to work on that terminal.

Input characters are always subjected to 1-to-1 mapping, even if they are part of a dead or compose key sequence.

Input characters must not be mapped to the NIL character (null byte); but a dead or compose key sequence may generate a NIL character, and a string to which an output character is mapped may include NIL characters.

No more than 255 dead and compose key sequences may be defined; and all the strings in the output section must not contain more than 255 characters in total.

The `ISTRIP` flag [see `stty(1)`] should not be set for a terminal to which a mapping table has been applied, as it causes the eighth bit to be stripped.

For the system administrator:

Each terminal map file requires a 2048-byte buffer in the operating system. If a number of terminals use the same map file, only one buffer is required, as the terminals then share the buffer. There is a limit on the number of buffers used for map files. The system administrator can configure this limit using the `NEMAP` system parameter [see `idtune(1M)`].

FILES

`/usr/lib/mapchan/*`

SEE ALSO

`stty(1)`, `mapchan(1M)`, `mapkey(1M)`, `ascii(5)`, `meta-ascii(5)`, `keyboard(7)` [on RM400], `tty(7)`.

System Administrator's Reference Manual

mdevice(4)

NAME

`mdevice -` driver configuration file

SYNOPSIS

`/etc/conf/cf.d/mdevice`

DESCRIPTION

The `mdevice` file is included in the `/etc/conf/cf.d` directory. It includes a one-line description of each device driver and configurable software module in the system to be built [except for file system types, see `mfsys(4)`]. Each line in `mdevice` represents the *Master* file component from a Driver Software Package (DSP) either delivered with the base system or installed later via `idinstall`.

Each line contains several whitespace-separated fields; they are described below. Each field must be supplied with a value or a "–" (dash).

1. *Device name*: This field is the internal name of the device or module, and may be up to 8 characters long. The first character of the name must be an alphabetic character; the others may be letters, digits, or underscores.
2. *Function list*: This field is a string of characters that identify driver functions that are present. Using one of the characters below requires the driver to have an entry point (function) of the type indicated. If no functions in the following list are supplied, the field should contain a dash.
   - `o–` open routine
   - `c–` close routine
   - `r–` read routine
   - `w–` write routine
   - `i–` ioctl routine
   - `s–` startup routine
   - `x–` exit routine
   - `f–` fork routine
   - `e–` exec routine
   - `I–` init routine
   - `h–` halt routine
   - `p–` poll routine
   - `L–` chpoll routine
   - `A–` supports `specaio` async interface
   - `B–` `specaio` breakup routine
   - `l–` latency routine
   - `d–` dump routine
   - `F–` flush routine
   - `C–` errcnt routine
   - `M–` character `mmap` function exists
   - `S–` character `segmap` function exists
   - `z–` size function exists
   - `P–` panic function exists

Note that if the device is a "block" type device (see field 3. below), a `strategy` routine and a `print` routine are required by default.

3. *Characteristics of driver*: This field contains a set of characters that indicate the characteristics of the driver. If none of the characters below apply, the field should contain a dash. The legal characters for this field are:
   - `i` – The device driver is installable.
   - `c` – The device is a "character" device.
   - `n` – The device driver is not installable.
   - `a` – The device driver is automatically installed.
   - `s` – Suppress the device count field.
   - `e` – Software exec module.
   - `m` – STREAMS module type.
   - `N` – No `driver.o` and `space.c`.
   - `R` – Driver has a reset routine.
   - `d` – Dispatcher class.
   - `f` – 4.0 based new driver.
   - `b` – The device is a "block" device.
   - `t` – The device is a tty.
   - `o` – This device may have only one `sdevice` entry.
   - `r` – This device is required in all configurations of the kernel. This option is intended for drivers delivered with the base system only.
   - `S` – This is a STREAMS module or driver.
   - `M` – This option indicates that the device requires multiple major numbers.
4. *Handler prefix*: This field contains the character string prepended to all the externally-known handler routines associated with this driver. The string may be up to 4 characters long.
5. *Block Major number*: If the device is a "block" type device, then this field should be set to the major number associated with the device. Otherwise, this field is ignored. For devices having multiple major numbers this field is used to specify the "range" of major numbers. (e.g. range 3-6 is interpreted as four major numbers between 3-6 inclusive).
6. *Character Major number*: If the device is a "character" type device or "STREAMS" type device, then this field should be set to the major number associated with this device. Otherwise, this field is ignored. For devices having multiple major numbers this field is used to specify the "range" of major numbers. (e.g. range 3-6 is interpreted as four major numbers between 3-6 inclusive).
7. *Minimum units*: This field is an integer specifying the minimum number of these devices that can be specified in the `sdevice` file.
8. *Maximum units*: This field specifies the maximum number of these devices that may be specified in the `sdevice` file. It contains an integer.
9. This field is not used in Reliant UNIX.

SPECIFYING STREAMS DEVICES AND MODULES

STREAMS modules and drivers are treated in a slightly different way from other drivers in all Reliant UNIX systems, and their configuration reflects this difference. To specify a STREAMS device driver, its `mdevice` entry should contain both an "S" and a "c" in the *characteristics* field (see 3. above). This indicates that it is a STREAMS driver and that it requires an entry in the Reliant UNIX kernel's `cdevsw` table, where STREAMS drivers are normally configured into the system.

A STREAMS module that is not a device driver, such as a line discipline module, requires an "`S`" and an "`m`" in the *characteristics* field of its `mdevice` file entry.

SEE ALSO

`idinstall`(1M), `mfsys`(4), `sdevice`(4).

System Administrator's Reference Manual

mfsys(4)

NAME

`mfsys` - file system configuration file

SYNOPSIS

`/etc/conf/cf.d/mfsys`

DESCRIPTION

The `mfsys` file contains configuration information for file system types that are to be included in the next system kernel to be built. It is located in the `/etc/conf/cf.d` directory and includes a one-line description of each file system type. The `mfsys` file is coalesced from component files in the `/etc/conf/mfsys.d` directory. Each line contains the following whitespace-separated fields:

1. *name*: This field contains the internal name for the file system type (e.g. `ufs`, `nfs`.) This name is no more than 32 characters long.
2. *prefix*: The *prefix* in this field is the string prepended to the *fstypsw* handler functions defined for this file system type (e.g. `s5`, `du`). The prefix must be no more that 8 characters long.
3. *flags*: The *flags* field contains a hex number of the form "0xNN" to be used in populating the *fsinfo* data structure table entry for this file system type.
4. *notify flags*: The *notify flags* field contains a hex number of the form "0xNN" to be used in population the *fsinfo* data structure table entry for this file system type.
5. *function bitstring*: The *function bitstring* is a string of 28 0's and 1's. Each file system type potentially defines 28 functions to populate the *fstypsw* data structure table entry for itself. All file system types do not supply all the functions in this table, however, and this bitstring is used to indicate which of the functions are present and which are absent. A "1" in this string indicates that a function has been supplied, and a "0" indicates that a function has not been supplied. Successive characters in the string represent successive elements of the *fstypsw* data structure, with the first entry in this data structure represented by the rightmost character in the string.

SEE ALSO

`idbuild`(1M), `idinstall`(1M), `sfsys`(4).

System Administrator's Reference Manual

mnttab(4)

NAME

mnttab - mounted file system table

SYNOPSIS

```
#include <sys/mnttab.h>
```

DESCRIPTION

The /etc/mnttab file contains information about devices that have been mounted by the mount(1M) command. The information is in the following structure, defined in <sys/mnttab.h>:

```
struct mnttab
{
    char *mnt_special;
    char *mnt_mountp;
    char *mnt_fstype;
    char *mnt_mntopts;
    char *mnt_time;
};
```

The fields in the mount table are space-separated and show the block special device, the mount point, the file system type of the mounted file system, the mount options, and the time at which the file system was mounted. The time at which the file system was mounted is specified decimally in seconds starting from 00:00:00 UTC (Coordinated Universal Time), January 1st, 1970. The mount(1M) command without a specified option displays this entry in the usual date format.

SEE ALSO

mount(1M), setmnt(1M), getmntent(3C).

System Administrator's Reference Manual

mtune(4)

NAME

`mtune` - tunable parameters (standard)

SYNOPSIS

`/etc/conf/cf.d/mtune`

DESCRIPTION

The `mtune` file contains information about all the system tunable parameters. Each tunable parameter is specified by a single line in the file, and each line contains the following whitespace-separated set of fields:

1. *parameter name*: A character string no more than 20 characters long. It is used to construct the preprocessor "#defines" that pass the value to the system when it is built.
2. *default value*: This is the default value of the tunable parameter. If the value is not specified in the `stune` file, this value will be used when the system is built. In addition to an integer value, this field may also contain a formula using other parameter names. Formulas may have parentheses, multiplication, division, addition and subtraction. Example: `(NPROC*10+5)`. White space is not allowed in a formula.
3. *minimum value*: This is the minimum allowable value for the tunable parameter. If the parameter is set in the `stune` file, the configuration tools will verify that the new value is equal to or greater than this value.
4. *maximum value*: This is the maximum allowable value for the tunable parameter. If the parameter is set in the `stune` file, the configuration tools will check that the new value is equal to or less than this value.

The `mtune` file normally resides in `/etc/conf/cf.d`. However, a user or an add-on package should never directly edit the `mtune` file to change the setting of a system tunable parameter. Instead the `idtune` command should be used to modify or append the tunable parameter to the `stune` file.

In order for the new values to become effective the Reliant UNIX system kernel must be rebuilt and the system must then be rebooted.

The `sysctl` interface provides another, more comfortable mechanism to influence the behavior of an "enhanced" system [cf. `protocol_stack(5)`] as is provided by the `idtune` tool. `sysctl(1M)` can be used to modify a running system; there is no need to link and reboot a new kernel for a change to be effective, but this functionality is essentially limited to networking tunables.

SEE ALSO

`idbuild(1M)`, `idtune(1M)`, `sysctl(1M)`, `stune(4)`, `protocol_stack(5)`.

System Administrator's Reference Manual

nologin(4)

NAME

`nologin -` the /etc/nologin file

DESCRIPTION

The `/etc/nologin` file can only be created by system administrators. Once this file has been created, users can no longer log in. This gives the system administrator the opportunity to carry out administration tasks, even in multiuser mode.

System Administrator's Reference Manual

osm.conf(4)

NAME

`osm.conf` - configuration file for the OSM osmd log daemon

SYNOPSIS

`/etc/default/osm.conf`

DESCRIPTION

The file contains three values which override the defaults for the `osmd` daemon (OSM – Operating System Messages). The configuration file is supplied in the following format:

```
# pathname of protocol files (without suffix):
PROTOCOL=/var/adm/log/osm
# number of files in cycle:
MAX=3
# size of protocol files in bytes (5MB):
SIZE=5120000
```

`PROTOCOL` defines the pathname of the protocol files. The messages are always written to the file without the suffix. As soon as this file has reached the maximum size permitted, a suffix is appended made up of date and time.

`MAX` defines the maximum number of files in the cycle. When a new protocol file has to be created, the oldest protocol files are deleted to ensure that the maximum number of files in the cycle is maintained again.

`SIZE` defines the maximum size per file.

The configuration file is not overwritten in the case of an update installation to ensure that manual changes to the settings are not lost. The signal 1 (`SIGHUP`) must be sent to the `osmd` daemon so that it can load the modified content of the `/etc/default/osm.conf` configuration file.

FILES

`/var/adm/log/osm`[`.`*MM-DD-hh:mm*]
protocol files

SEE ALSO

`osmd`(1M) .

ott(4)

NAME

`ott` - FACE object architecture information

DESCRIPTION

The FACE object architecture stores information about object types in an ASCII file named `.ott` (object type table) that is contained in each directory. This file describes all of the objects in that directory. Each line of the `.ott` file contains information about one object in pipe-separated fields. The fields are (in order):

name The name of the actual Reliant UNIX system file.

dname
The name that should be displayed to the user, or a dot if it is the same as the name of the file.

description
The description of the object, or a dot if the description is the default (the same as object type).

object-type
The FACE internal object type name.

flags
Object specific flags.

mod time
The time that FACE last modified the object. The time is given as number of seconds since 1/1/1970, and is in hexadecimal notation.

object information
An optional field, contains a set of semi-colon separated *name=value* fields that can be used by FACE to store any other information necessary to describe this object.

FILES

`.ott` is created in any directory opened by FACE.

passwd(4) - vfstab(4)

passwd(4)

NAME

`passwd` - password file

DESCRIPTION

The `/etc/passwd` file contains the following information for each user:

- login name
- dummy password
- numerical user ID
- numerical group ID
- comment
- initial working directory
- program to use as shell

`passwd` is an ASCII file. Each field within each user's entry is separated from the next by a colon. The comment field can contain any desired information. Each user is separated from the next by a new line. If the shell field is null, `/usr/bin/sh` is used.

This file has user login information and general read permission. It can therefore be used, for example, to map numerical user IDs to names.

The *password* field consists of the character `x`. This field remains only for compatibility reasons. Password information is contained in the `/etc/shadow` file [see shadow(4)].

FILES

`/etc/passwd`

`/etc/shadow`

SEE ALSO

login(1), passwd(1), pwconv(1M), useradd(1M), userdel(1M), usermod(1M), a64l(3C), getpwent(3C), putpwent(3C), group(4), shadow(4), unistd(4).

System Administrator's Reference Manual

pathalias(4)

NAME

`pathalias -` alias file for FACE

DESCRIPTION

The `pathalias` files contain lines of the form `alias=`*path* where *path* can be one or more colon-separated directories. Whenever a FACE user references a path not beginning with a "`/`", this file is checked. If the first component of the pathname matches the left-hand side of the equals sign, the right-hand side is searched much like `$PATH` variable in the Reliant UNIX system. This allows users to reference the folder `$HOME/FILECABINET` by typing `filecabinet`.

There is a system-wide `pathalias` file called `$VMSYS/pathalias`, and each user can also have local alias file called `$HOME/pref/pathalias`. Settings in the user alias file override settings in the system-wide file. The system-wide file is shipped with several standard FACE aliases, such as `filecabinet`, `wastebasket`, `preferences`, `other_users`, etc.

NOTES

Unlike command keywords, partial matching of a path alias is not permitted, however, path aliases are case insensitive. The name of an alias should be alphabetic, and in no case can it contain special characters like "`/`", "`\`", or "`=`". There is no particular limit on the number of aliases allowed. Alias files are read once, at login, and are held in core until logout. Thus, if an alias file is modified during a session, the change will not take effect until the next session.

FILES

`$HOME/pref/pathalias`
`$VMSYS/pathalias`

System Administrator's Reference Manual

pkginfo(4)

NAME

pkginfo - package characteristics file

DESCRIPTION

pkginfo is an ASCII file that describes the characteristics of the package along with information that helps control the flow of installation. It is created by the software package developer.

Each entry in the pkginfo file is a line that establishes the value of a parameter in the following form:

PARAM="*value*"

There is no required order in which the parameters must be specified within the file. Each parameter is described below. Only fields marked with an asterisk (*) are mandatory.

PKG* Abbreviation for the package being installed, generally three characters in length (for example, dir or pkg). All characters in the abbreviation must be alphanumeric and the first must not be numeric. The abbreviation is limited to a maximum length of nine characters. install, new, and all are reserved abbreviations.

NAME*
: Text that specifies the package name (maximum length of 256 ASCII characters).

ARCH*
: A comma-separated list of alphanumeric tokens that indicate the architecture (for example, MIPS) associated with the package. The pkgmk tool may be used to create or modify this value when actually building the package. The maximum length of a token is characters and it cannot include a comma. This value can be specified with the –a option of the pkgmk command when the package is being built.

VERSION*
: Text that specifies the current version associated with the software package. The maximum length is 256 ASCII characters and the first character cannot be a left parenthesis. The pkgmk tool may be used to create or modify this value when actually building the package.

CATEGORY*
: A comma-separated list of categories under which a package may be displayed. A package must at least belong to the system or application category. Categories are case-insensitive and may contain only alphanumerics. Each category is limited in length to characters.

DESC Text that describes the package (maximum length of 256 ASCII characters).

VENDOR
: Used to identify the vendor who holds the software copyright (maximum length of 256 ASCII characters).

HOTLINE
: Phone number and/or mailing address where further information may be received or bugs may be reported (maximum length of 256 ASCII characters).

EMAIL
: An electronic address where further information is available or bugs may be reported (maximum length of 256 ASCII characters).

VSTOCK
: The vendor stock number, if any, that identifies this product (maximum length of 256 ASCII characters).

CLASSES
: A space-separated list of classes defined for a package. The order of the list determines the order in which the classes are installed. Classes listed first will be installed first (on a media by media basis). This parameter may be modified by the request script.

ISTATES
: A list of permissible run states for package installation (for example, "S s 1").

RSTATES
: A list of permissible run states for package removal (for example, "S s 1").

BASEDIR
: The standard basic directory for collectively relocatable objects in the package. If package objects are to be freely relocatable, this parameter must be present and have a value other than zero.

ULIMIT
: If set, this parameter is passed as an argument to the ulimit command, which establishes the maximum size of a file during installation.

ORDER
: A list of classes defining the order in which they should be put on the medium. Used by pkgmk in creating the package. Classes not defined in this field are placed on the medium using the standard ordering procedures.

MAXINST
: The maximum number of package instances that should be allowed on a machine at the same time. By default, only one instance of a package is allowed. This parameter must be set in order to have multiple instances of a package.

PSTAMP
: Production stamp used to mark the pkgmap file on the output volumes. Provides a means of distinguishing between production copies of a version if more than one is in use at a time. If PSTAMP is not defined, the default is used. The default consists of the Reliant UNIX system machine name followed by the string "*YYMMDDHHMM*" (year, month, date, hour, minutes).

INTONLY
: Indicates that the package should only be installed interactively when set to any non-zero value.

PREDEPEND
: Used to maintain compatibility with pre-SVR4 package dependency checking. Pre-SVR4 dependency checks were based on whether or not the name file for the required package existed in the /var/options directory. This directory is not maintained for SVR4 packages since the depend file is used for checking dependencies. However, entries can be created in this directory to maintain compatibility. Setting the PREDEPEND parameter to y or yes creates a /usr/option entry for the package. (Packages that are new for SVR4 do not need to use this parameter.)

EXAMPLES

Here is a sample pkginfo:

```
PKG="oam"
NAME="OAM Installation Utilities"
VERSION="1"
VENDOR="AT&T"
HOTLINE="1-800-ATT-BUGS"
EMAIL="attunix!olsen"
VSTOCK="0122c3f5566"
CATEGORY="system.essential"
ISTATES="S 2"
RSTATES="S 2"
```

NOTES

Developers may define their own installation parameters by adding a definition to this file. A developer-defined parameter must begin with an uppercase letter.

The parameter values can be enclosed in single or double quotes (' or "). Use of these two characters should be avoided within the values, alternatively they can be preceded by \ (this character is retained).

System Administrator's Reference Manual

pkgmap(4)

NAME

`pkgmap` - package contents description file

DESCRIPTION

`pkgmap` is an ASCII file that provides a complete listing of the package contents. It is automatically generated by `pkgmk(1M)` using the information in the `prototype` file.

Each entry in `pkgmap` describes a single "deliverable object file". A deliverable object file includes shell scripts, executable objects, data files, directories, etc. The entry consists of several fields of information, each field separated by a space. The fields are described below and must appear in the order shown.

part An optional field designating the part number in which the object resides. A part is a collection of files, and is the atomic unit by which a package is processed. A developer can choose the criteria for grouping files into a part (e.g. based on class). If no value is defined in this field, part 1 is assumed.

ftype

A one-character field that indicates the file type. Valid values are:

- `f` astandard executable or data file
- `e` afile to be edited upon installation or removal
- `v` volatile file (one whose contents are expected to change)
- `d` directory
- `x` an exclusive directory
- `l` linked file
- `p` named pipe
- `c` character special device
- `b` block special device
- `i` installation script or information file
- `s` symbolic link

class

The installation class to which the file belongs. This name must contain only alphanumeric characters and be no longer than 12 characters. It is not specified if the *ftype* is `i` (information file).

pathname

The pathname where the object will reside on the target machine, such as `/usr/bin/mail`. Relative pathnames (those that do not begin with a slash) indicate that the file is relocatable.

For linked files (*ftype* is either `l` or `s`), pathname must be in the form of *path1=path2*, with *path1* specifying the destination of the link and *path2* specifying the source of the link.

pathname may contain variables which support relocation of the file. A `$`*parameter* may be embedded in the pathname structure. `$BASEDIR` can be used to identify the parent directories of the path hierarchy, making the entire package easily relocatable. Default values for *parameter* and `BASEDIR` must be supplied in the `pkginfo` file and may be overridden at installation.

major

The major device number. The field is only specified for block or character special devices.

minor

The minor device number. The field is only specified for block or character special devices.

mode The octal mode of the file (for example, 0664). A question mark (?) indicates that the mode will be left unchanged, implying that the file already exists on the target machine. This field is not used for linked

files, packaging information files or non-installable files.

owner

The owner of the file (for example, `bin` or `root`). The field is limited to 14 characters in length. A question mark (?) indicates that the owner will be left unchanged, implying that the file already exists on the target machine. This field is not used for linked files or non-installable files. It is used optionally with a package information file. If used, it indicates with what owner an installation script will be executed.

Can be a variable specification in the form of `$[A-Z]`. Will be resolved at installation time.

group

The group to which the file belongs (for example, `bin` or `sys`). The field is limited to 14 characters in length. A question mark (?) indicates that the group will be left unchanged, implying that the file already exists on the target machine. This field is not used for linked files or non-installable files. It is used optionally with a package information file. If used, it indicates with what group an installation script will be executed.

Can be a variable assignment in the form of `$[A-Z]`. Will be resolved at installation time.

size The actual size of the file in bytes. This field is not specified for named pipes, special devices, directories or linked files.

cksum

The checksum of the file contents. This field is not specified for named pipes, special devices, directories or linked files.

modtime

The time of last modification, as reported by the `stat(2)` function call. This field is not specified for named pipes, special devices, directories or linked files.

Each `pkgmap` must have one line that provides information about the number and maximum size (in 512-byte blocks) of parts that make up the package. This line is in the following format:

`:`*number_of_parts maximum_part_size*

Lines that begin with # are comment lines and are ignored.

When files are saved during installation before they are overwritten, they are normally just copied to a temporary pathname. However, for files whose mode includes execute permission (but which are not editable), the existing version is linked to a temporary pathname and the original file is removed. This allows processes which are executing during installation to be overwritten.

EXAMPLES

The following is an example of a `pkgmap` file.

```
:2 500
1 i pkginfo 237 1179 541296672
1 b class1 /dev/ios0/stape0004 17 134 0644 root other
1 c class1 /dev/ios0/stape0004 17 134 0644 root other
1 d none bin 0755 root bin
1 f none bin/INSTALL 0755 root bin 11103 17954 541295535
1 f none bin/REMOVE 0755 root bin 3214 50237 541295541
1 l none bin/UNINSTALL=bin/REMOVE
1 f none bin/cmda 0755 root bin 3580 60325 541295567
1 f none bin/cmdb 0755 root bin 49107 51255 541438368
1 f class1 bin/cmdc 0755 root bin 45599 26048 541295599
1 f class1 bin/cmdd 0755 root bin 4648 8473 541461238
1 f none bin/cmde 0755 root bin 40501 1264 541295622
1 f class2 bin/cmdf 0755 root bin 2345 35889 541295574
1 f none bin/cmdg 0755 root bin 41185 47653 541461242
2 d class2 data 0755 root bin
2 p class1 data/apipe 0755 root other
2 d none log 0755 root bin
```

```
2 v none log/logfile 0755 root bin 41815 47563 541461333
2 d none save 0755 root bin
2 d none spool 0755 root bin
2 d none tmp 0755 root bin
```

NOTES

The `pkgmap` file may contain only one entry per unique pathname.

SEE ALSO

`pkgmk`(1M), `prototype`(4).

System Administrator's Reference Manual

pnch(4)

NAME

`pnch` - file format for card images

DESCRIPTION

The PNCH format is a convenient representation for files consisting of card images in an arbitrary code.

A PNCH file is a simple concatenation of card records. A card record consists of a single control byte followed by a variable number of data bytes. The control byte specifies the number (which must lie in the range 0–80) of data bytes that follow. The data bytes are 8-bit codes that constitute the card image. If there are fewer than 80 data bytes, it is understood that the remainder of the card image consists of trailing blanks.

System Administrator's Reference Manual

proc(4)

NAME

`proc` - /proc process file system

DESCRIPTION

`/proc` is a file system that provides access to the image of each active process in the system. The name of each entry in the `/proc` directory is a decimal number corresponding to the process ID. The owner of each "file" is determined by the process's user ID.

Standard system call interfaces are used to access `/proc` files: `open`, `close`, `read`, `write`, and `ioctl`. An open for reading and writing enables process control; a read-only open allows inspection but not control. As with ordinary files, more than one process can open the same `/proc` file at the same time. Exclusive open is provided to allow controlling processes to avoid collisions: an `open` for writing that specifies `O_EXCL` fails if the file is already open for writing; if such an exclusive open succeeds, subsequent attempts to open the file for writing, with or without the `O_EXCL` flag, fail until the exclusively-opened file descriptor is closed. (Exception: a superuser open that does not specify `O_EXCL` succeeds even if the file is exclusively opened.) There can be any number of read-only opens, even when an exclusive write open is in effect on the file.

Data may be transferred from or to any locations in the traced process's address space by applying `lseek` to position the file at the virtual address of interest followed by `read` or `write`. The `PIOCMAP` operation can be applied to determine the accessible areas (mappings) of the address space. A contiguous area of the address space may appear as multiple mappings due to varying read/write/execute permissions. I/O transfers may span contiguous mappings. An I/O request extending into an unmapped area is truncated at the boundary.

Information and control operations are provided through `ioctl`. These have the form:

```
#include <sys/types.h>
#include <sys/signal.h>
#include <sys/fault.h>
#include <sys/syscall.h>
#include <sys/procfs.h>
void *p;
retval = ioctl(fildes, code, p);
```

The argument *p* is a generic pointer whose type depends on the specific `ioctl` code. Where not specifically mentioned below, its value should be zero. `<sys/procfs.h>` contains definitions of `ioctl` codes and data structures used by the operations. Certain operations can be performed only if the process file is open for writing; these include all operations that affect process control.

Process information and control operations involve the use of sets of flags. The set types `sigset_t`, `fltset_t`, and `sysset_t` correspond, respectively, to signal, fault, and system call enumerations defined in `<sys/signal.h>`, `<sys/fault.h>`, and `<sys/syscall.h>`.

Each set type is large enough to hold flags for its own enumeration. Although they are of different sizes, they have a common structure and can be manipulated by these macros:

```
prfillset(&set);            /* turn on all flags in set */
premptyset(&set);           /* turn off all flags in set */
praddset(&set, flag);       /* turn on the specified flag */
prdelset(&set, flag);       /* turn off the specified flag */
r = prismember(&set, flag); /* != 0 iff flag is turned on */
```

One of `prfillset` or `premptyset` must be used to initialize `set` before it is used in any other operation. `flag` must be a member of the enumeration corresponding to `set`.

The allowable `ioctl` codes follow. Those requiring write access are marked with an asterisk (*). Except where noted, an `ioctl` to a process that has terminated elicits the error `ENOENT`.

PIOCSTATUS

This returns status information for the process; *p* is a pointer to a `prstatus` structure:

```
typedef struct prstatus
{
  long        pr_flags;          /* Process flags */
  short       pr_why;            /* Reason for process stop (if stopped) */
  short       pr_what;           /* More detailed reason */
  struct siginfo     pr_info;    /* Info associated with signal or fault */
  short       pr_cursig;         /* Current signal */
  sigset_t    pr_sigpend;        /* Set of other pending signals */
  sigset_t    pr_sighold;        /* Set of held signals */
  struct sigaltstack pr_altstack; /* Alternate signal stack info */
  struct sigaction   pr_action;  /* Signal action for current signal */
  pid_t       pr_pid;            /* Process id */
  pid_t       pr_ppid;           /* Parent process id */
  pid_t       pr_pgrp;           /* Process group id */
  pid_t       pr_sid;            /* Session id */
  timestruc_t pr_utime;          /* Process user cpu time */
  timestruc_t pr_stime;          /* Process system cpu time */
  timestruc_t pr_cutime;         /* Sum of children's user times */
  timestruc_t pr_cstime;         /* Sum of children's system times */
  char        pr_clname[8];      /* Scheduling class name */
  long        pr_filler[20];     /* Filler area for future expansion */
  long        pr_instr;          /* Current instruction */
  gregset_t   pr_reg;            /* General registers */
} prstatus_t;
```

`pr_flags` is a bit-mask holding these flags:

`PR_STOPPED`
: process is stopped

`PR_ISTOP`
: process is stopped on an event of interest (see `PIOCSTOP`)

`PR_DSTOP`
: process has a stop directive in effect (see `PIOCSTOP`)

`PR_ASLEEP`
: process is in an interruptible sleep within a system call

`PR_FORK`
: process has its inherit-on-`fork` flag set (see `PIOCSFORK`)

`PR_RLC`
: process has its run-on-last-`close` flag set (see `PIOCSRLC`)

`PR_PTRACE`
: process is being traced via `ptrace`

`PR_PCINVAL`
: process program counter refers to an invalid address

`PR_ISSYS`
: process is a system process (see `PIOCSTOP`)

`pr_why` and `pr_what` together describe, for a stopped process, the reason that the process is stopped. Possible values of `pr_why` are:

- `PR_REQUESTED` indicates that the process stopped because `PIOCSTOP` was applied; `pr_what` is unused in this case.
- `PR_SIGNALLED` indicates that the process stopped on receipt of a signal (see `PIOCSTRACE`); `pr_what` holds the signal number that caused the stop (for a newly-stopped process, the same value is in `pr_cursig`).
- `PR_FAULTED` indicates that the process stopped on incurring a hardware fault (see `PIOCSFAULT`); `pr_what` holds the fault number that caused the stop.
- `PR_SYSENTRY` and `PR_SYSEXIT` indicate a stop on entry to or exit from a system call (see `PIOCSENTRY`

and PIOCSEXIT); pr_what holds the system call number.

- PR_JOBCONTROL indicates that the process stopped due to the default action of a job control stop signal [see sigaction(2)]; pr_what holds the stopping signal number.

pr_info, when the process is in a PR_SIGNALLED or PR_FAULTED stop, contains additional information pertinent to the particular signal or fault (see <sys/siginfo.h>).

pr_cursig names the current signal – that is, the next signal to be delivered to the process. pr_sigpend identifies any other pending signals. pr_sighold identifies those signals whose delivery is being delayed if sent to the process.

pr_altstack contains the alternate signal stack information for the process [see sigaltstack(2)]. pr_action contains the signal action information pertaining to the current signal [see sigaction(2)]; it is undefined if pr_cursig is zero.

pr_pid, pr_ppid, pr_pgrp, and pr_sid are, respectively, the process id, the id of the process's parent, the process's process group id, and the process's session id.

pr_utime, pr_stime, pr_cutime, and pr_cstime are, respectively, the user CPU and system CPU time consumed by the process, and the cumulative user CPU and system CPU time consumed by the process's children, in seconds and nanoseconds.

pr_clname contains the name of the process's scheduling class.

The pr_filler area is reserved for future use.

pr_instr contains the machine instruction to which the program counter refers. The amount of data retrieved from the process is machine-dependent; on the 3B2, it is a single byte. In general, the size is that of the machine's smallest instruction. If the program counter refers to an invalid address, PR_PCINVAL is set and pr_instr is undefined.

pr_reg is an array holding the contents of the general registers. On the 3B2 the predefined constants R_R0, R_R1, ... R_R8, R_FP, R_AP, R_PS, R_SP, and R_PC can be used as indices to refer to the corresponding registers.

PIOCSTOP*, PIOCWSTOP

PIOCSTOP directs the process to stop and waits until it has stopped; PIOCWSTOP simply waits for the process to stop. These operations complete when the process stops on an event of interest, immediately if already so stopped. If *p* is non-zero it points to an instance of prstatus_t to be filled with status information for the stopped process.

An "event of interest" is either a PR_REQUESTED stop or a stop that has been specified in the process's tracing flags (set by PIOCSTRACE, PIOCSFAULT, PIOCSENTRY, and PIOCSEXIT). A PR_JOBCONTROL stop is specifically not an event of interest. (A process may stop twice due to a stop signal, first showing PR_SIGNALLED if the signal is traced and again showing PR_JOBCONTROL if the process is set running without clearing the signal.) If the process is controlled by ptrace, it comes to a PR_SIGNALLED stop on receipt of any signal; this is an event of interest only if the signal is in the traced signal set. If PIOCSTOP is applied to a process that is stopped, but not on an event of interest, the stop directive takes effect when the process is restarted by the competing mechanism; at that time the process enters a PR_REQUESTED stop before executing any user-level code.

ioctls are interruptible by signals so that, for example, an alarm(2) can be set to avoid waiting forever for a process that may never stop on an event of interest. If PIOCSTOP is interrupted, the stop directive remains in effect even though the ioctl returns an error.

A system process (indicated by the PR_ISSYS flag) never executes at user level, has no user-level address space visible through /proc, and cannot be stopped. Applying PIOCSTOP or PIOCWSTOP to a system process elicits the error EBUSY.

PIOCRUN*

The traced process is made runnable again after a stop. If *p* is non-zero it points to a prrun structure describing additional actions to be performed:

```
typedef struct prrun
{
    long      pr_flags;     /* Flags */
    sigset_t  pr_trace;     /* Set of signals to be traced */
    sigset_t  pr_sighold;   /* Set of signals to be held */
    fltset_t  pr_fault;     /* Set of faults to be traced */
    caddr_t   pr_vaddr;     /* Virtual address at which to resume */
    long      pr_filler[8]; /* Filler area for future expansion */
} prrun_t;
```

`pr_flags` is a bit-mask describing optional actions; the remainder of the entries are meaningful only if the appropriate bits are set in `pr_flags`. `pr_filler` is reserved for future use; this area must be filled with zeros by the user's program. Flag definitions:

- `PRCSIG` clears the current signal, if any (see `PIOCSSIG`).
- `PRCFAULT` clears the current fault, if any (see `PIOCCFAULT`).
- `PRSTRACE` sets the traced signal set to `pr_trace` (see `PIOCSTRACE`).
- `PRSHOLD` sets the held signal set to `pr_sighold` (see `PIOCSHOLD`).
- `PRSFAULT` sets the traced fault set to `pr_fault` (see `PIOCSFAULT`).
- `PRSVADDR` sets the address at which execution resumes to `pr_vaddr`.
- `PRSTEP` directs the process to single-step – i.e. to run and to execute a single machine instruction. On completion of the instruction, a hardware trace trap occurs. If `FLTTRACE` is being traced, the process stops, otherwise it is sent `SIGTRAP`; if `SIGTRAP` is being traced and not held, the process stops. This operation requires hardware support and may not be implemented on all processors.
- `PRSABORT` is meaningful only if the process is in a `PR_SYSENTRY` stop or is marked `PR_ASLEEP`; it instructs the process to abort execution of the system call (see `PIOCSENTRY`, `PIOCSEXIT`).
- `PRSTOP` directs the process to stop again as soon as possible after resuming execution (see `PIOCSTOP`). In particular if the process is stopped on `PR_SIGNALLED` or `PR_FAULTED`, the next stop will show `PR_REQUESTED`, no other stop will have intervened, and the process will not have executed any user-level code.

`PIOCRUN` fails (`EBUSY`) if applied to a process that is not stopped on an event of interest. Once `PIOCRUN` has been applied, the process is no longer stopped on an event of interest even if, due to a competing mechanism, it remains stopped.

PIOCSTRACE*

This defines a set of signals to be traced: the receipt of one of these signals causes the traced process to stop. The set of signals is defined via an instance of `sigset_t` addressed by *p*. Receipt of `SIGKILL` cannot be traced.

If a signal that is included in the held signal set is sent to the traced process, the signal is not received and does not cause a process stop until it is removed from the held signal set, either by the process itself or by setting the held signal set with `PIOCSHOLD` or the `PRSHOLD` option of `PIOCRUN`.

PIOCGTRACE

The current traced signal set is returned in an instance of `sigset_t` addressed by *p*.

PIOCSSIG*

The current signal and its associated signal information are set according to the contents of the `siginfo` structure addressed by *p* (see `<sys/siginfo.h>`). If the specified signal number is zero or if *p* is zero, the current signal is cleared. The semantics of this operation are different from those of `kill` or `PIOCKILL` in that the signal is delivered to the process immediately after execution is resumed (even if it is being held) and an additional `PR_SIGNALLED` stop does not intervene even if the signal is traced. Setting the current signal to `SIGKILL` terminates the process immediately, even if it is stopped.

PIOCKILL*

A signal is sent to the process with semantics identical to those of `kill`; *p* points to an `int` naming the signal. Sending `SIGKILL` terminates the process immediately.

PIOCUNKILL*

A signal is deleted, i.e. it is removed from the set of pending signals; the current signal (if any) is unaffected. *p* points to an `int` naming the signal. It is an error to attempt to delete `SIGKILL`.

PIOCGHOLD, PIOCSHOLD*

`PIOCGHOLD` returns the set of held signals (signals whose delivery will be delayed if sent to the process) in an instance of `sigset_t` addressed by *p*. `PIOCSHOLD` correspondingly sets the held signal set but does not allow `SIGKILL` or `SIGSTOP` to be held.

PIOCMAXSIG, PIOCACTION

These operations provide information about the signal actions associated with the traced process [see `sigaction`(2)]. `PIOCMAXSIG` returns, in the `int` addressed by *p*, the maximum signal number understood by the system. This can be used to allocate storage for use with the `PIOCACTION` operation, which returns the traced process's signal actions in an array of `sigaction` structures addressed by *p*. Signal numbers are displaced by 1 from array indices, so that the action for signal number *n* appears in position *n*–1 of the array.

PIOCSFAULT*

This defines a set of hardware faults to be traced: on incurring one of these faults the traced process stops. The set is defined via an instance of `fltset_t` addressed by *p*. Fault names are defined in `<sys/fault.h>` and include the following. Some of these may not occur on all processors; there may be processor-specific faults in addition to these.

`FLTILL`
: illegal instruction

`FLTPRIV`
: privileged instruction

`FLTBPT`
: breakpoint trap

`FLTTRACE`
: trace trap

`FLTACCESS`
: memory access fault

`FLTBOUNDS`
: memory bounds violation

`FLTIOVF`
: integer overflow

`FLTIZDIV`
: integer zero divide

`FLTFPE`
: floating-point exception

`FLTSTACK`
: unrecoverable stack fault

`FLTPAGE`
: recoverable page fault

When not traced, a fault normally results in the posting of a signal to the process that incurred the fault. If the process stops on a fault, the signal is posted to the process when execution is resumed unless the fault is cleared by `PIOCCFAULT` or by the `PRCFAULT` option of `PIOCRUN`. `FLTPAGE` is an exception; no signal is posted. There may be additional processor-specific faults like this. `pr_info` in the `prstatus` structure identifies the signal to be sent and contains machine-specific information about the fault.

PIOCGFAULT

The current traced fault set is returned in an instance of `fltset_t` addressed by *p*.

PIOCCFAULT*

The current fault (if any) is cleared; the associated signal is not sent to the process.

PIOCSENTRY*, PIOCSEXIT*

These operations instruct the process to stop on entry to or exit from specified system calls. The set of system calls to be traced is defined via an instance of `sysset_t` addressed by *p*.

When entry to a system call is being traced, the traced process stops after having begun the call to the system but before the system call arguments have been fetched from the process. When exit from a system call is being traced, the traced process stops on completion of the system call just prior to checking for signals and returning to user level. At this point all return values have been stored into the traced process's saved registers.

If the traced process is stopped on entry to a system call (`PR_SYSENTRY`) or when sleeping in an interruptible system call (`PR_ASLEEP` is set), it may be instructed to go directly to system call exit by specifying the `PRSABORT` flag in a `PIOCRUN` request. Unless exit from the system call is being traced the process returns to user level showing error `EINTR`.

PIOCGENTRY, PIOCGEXIT

These return the current traced system call entry or exit set in an instance of `sysset_t` addressed by *p*.

PIOCSFORK*, PIOCRFORK*

`PIOCSFORK` sets the inherit-on-fork flag in the traced process: the process's tracing flags are inherited by the child of a `fork`. `PIOCRFORK` turns this flag off: child processes start with all tracing flags cleared.

PIOCSRLC*, PIOCRRLC*

`PIOCSRLC` sets the run-on-last-close flag in the traced process: when the last writable `/proc` file descriptor referring to the traced process is closed, all of the process's tracing flags are cleared, any outstanding stop directive is canceled, and if the process is stopped, it is set running as though `PIOCRUN` had been applied to it. `PIOCRRLC` turns this flag off: the process's tracing flags are retained and the process is not set running when the process file is closed.

PIOCGREG, PIOCSREG*

These operations respectively get and set the saved process registers into or out of an array addressed by *p*; the array has type `gregset_t`. Register contents are accessible using a set of predefined indices (see `PIOCSTATUS`). Only certain bits of the processor-status word (PSW) can be modified by `PIOCSREG`; on the 3B2 these include the condition-code bits and the trace-enable bit. Other privileged registers cannot be modified at all. `PIOCSREG` fails (`EBUSY`) if applied to a process that is not stopped on an event of interest.

PIOCGFPREG, PIOCSFPREG*

These operations respectively get and set the saved process floating-point registers into or out of a structure addressed by *p*; the structure has type `fpregset_t`. An error (`EINVAL`) is returned if there is no floating-point hardware on the machine. `PIOCSFPREG` fails (`EBUSY`) if applied to a process that is not stopped on an event of interest.

PIOCNICE*

The traced process's `nice(1)` priority is incremented by the amount contained in the `int` addressed by *p*. Only the superuser may better a process's priority in this way, but any user may make the priority worse.

PIOCPSINFO

This returns miscellaneous process information such as that reported by `ps(1)` . *p* is a pointer to a `prpsinfo`

structure containing at least the following fields:

```
typedef struct prpsinfo
{
    char    pr_state;  /* numeric process state (see pr_sname) */
    char    pr_sname;  /* printable character representing pr_state */
    char    pr_zomb;   /* !=0: process terminated but not waited for */
    char    pr_nice;   /* nice for cpu usage */
    u_long  pr_flag;   /* process flags */
    uid_t   pr_uid;    /* real user id */
    gid_t   pr_gid;    /* real group id */
    pid_t   pr_pid;    /* unique process id */
    pid_t   pr_ppid;   /* process id of parent */
    pid_t   pr_pgrp;   /* pid of process group leader */
    pid_t   pr_sid;    /* session id */
    caddr_t pr_addr;   /* physical address of process */
    long    pr_size;   /* size of process image in pages */
    long    pr_rssize; /* resident set size in pages */
    caddr_t pr_wchan;  /* wait addr for sleeping process */
    timestruc_t pr_start;  /* process start time, sec+nsec since epoch */
    timestruc_t pr_time;   /* usr+sys cpu time for this process */
    long    pr_pri;        /* priority, high value is high priority */
    char    pr_oldpri;     /* pre-V5.41, low value is high priority */
    char    pr_cpu;        /* pre-V5.41, cpu usage for scheduling */
    dev_t   pr_ttydev;     /* controlling tty device (PRNODEV if none) */
    char    pr_clname[8];  /* Scheduling class name */
    char    pr_fname[16];  /* last component of execed pathname */
    char    pr_psargs[PRARGSZ]; /* initial characters of arg list */
    long    pr_filler[20]; /* for future expansion */
} prpsinfo_t;
```

Some of the entries in `prpsinfo`, such as `pr_state` and `pr_flag`, are system-specific and should not be expected to retain their meanings across different versions of the operating system. `pr_addr` is a vestige of the past and has no real meaning in current systems.

`PIOCPSINFO` can be applied to a zombie process (one that has terminated but whose parent has not yet performed a `wait` on it).

PIOCNMAP, PIOCMAP

These operations provide information about the memory mappings (virtual address ranges) associated with the traced process. `PIOCNMAP` returns, in the `int` addressed by *p*, the number of mappings that are currently active. This can be used to allocate storage for use with the `PIOCMAP` operation, which returns the list of currently active mappings. For `PIOCMAP`, *p* addresses an array of elements of type `prmap_t`; one array element (one structure) is returned for each mapping, plus an additional element containing all zeros to mark the end of the list.

```
typedef struct prmap
{
    caddr_t  pr_vaddr;      /* Virtual address base */
    u_long   pr_size;       /* Size of mapping in bytes */
    off_t    pr_off;        /* Offset into mapped object, if any */
    long     pr_mflags;     /* Protection and attribute flags */
    long     pr_filler[4];  /* Filler for future expansion */
} prmap_t;
```

`pr_vaddr` is the virtual address base (the lower limit) of the mapping within the traced process and `pr_size` is its size in bytes. `pr_off` is the offset within the mapped object (if any) to which the address base is mapped.

`pr_mflags` is a bit-mask of protection and attribute flags:

`MA_READ` mapping is readable by the traced process

MA_WRITE
: mapping is writable by the traced process

MA_EXEC
: mapping is executable by the traced process

MA_SHARED
: mapping changes are shared by the mapped object

MA_BREAK
: mapping is grown by the brk system call

MA_STACK
: mapping is grown automatically on stack faults

PIOCOPENM

The return value *retval* provides a read-only file descriptor for a mapped object associated with the traced process. If *p* is zero the traced process's execed file (its a.out file) is found. This enables a debugger to find the object file symbol table without having to know the path name of the executable file. If *p* is non-zero it points to a caddr_t containing a virtual address within the traced process and the mapped object, if any, associated with that address is found; this can be used to get a file descriptor for a shared library that is attached to the process. On error (invalid address or no mapped object for the designated address), −1 is returned.

PIOCCRED

Fetch the set of credentials associated with the process. *p* points to an instance of prcred_t, which is filled by the operation:

```
typedef struct prcred
{
    uid_t    pr_euid;     /* Effective user id */
    uid_t    pr_ruid;     /* Real user id */
    uid_t    pr_suid;     /* Saved user id (from exec) */
    uid_t    pr_egid;     /* Effective group id */
    uid_t    pr_rgid;     /* Real group id */
    uid_t    pr_sgid;     /* Saved group id (from exec) */
    u_int    pr_ngroups;  /* Number of supplementary groups */
} prcred_t;
```

PIOCGROUPS

Fetch the set of supplementary group IDs associated with the process. *p* points to an array of elements of type uid_t, which will be filled by the operation. PIOCCRED can be applied beforehand to determine the number of groups (pr_ngroups) that will be returned and the amount of storage that should be allocated to hold them.

PIOCGETPR, PIOCGETU

These operations copy, respectively, the traced process's proc structure and user area into the buffer addressed by *p*. They are provided for completeness but it should be unnecessary to access either of these structures directly since relevant status information is available through other control operations. Their use is discouraged because a program making use of them is tied to a particular version of the operating system.

PIOCGETPR can be applied to a zombie process (see PIOCPSINFO).

DIAGNOSTICS

Errors that can occur in addition to the errors normally associated with file system access:

ENOENT
: The traced process has exited after being opened.

EIO I/O was attempted at an illegal address in the traced process.

EBADF
: An I/O or ioctl operation requiring write access was attempted on a file descriptor not open for

writing.

EBUSY
: PIOCSTOP or PIOCWSTOP was applied to a system process; an exclusive open was attempted on a process file already already open for writing; an open for writing was attempted and an exclusive open is in effect on the process file; PIOCRUN, PIOCSREG or PIOCSFPREG was applied to a process not stopped on an event of interest; an attempt was made to mount /proc when it is already mounted.

EPERM
: Someone other than the superuser attempted to better a process's priority by issuing PIOCNICE.

ENOSYS
: An attempt was made to perform an unsupported operation (such as create, remove, link, or unlink) on an entry in /proc.

EFAULT
: An I/O or ioctl request referred to an invalid address in the controlling process.

EINVAL
: In general this means that some invalid argument was supplied to a system call. The list of conditions eliciting this error includes: the ioctl code is undefined; an ioctl operation was issued on a file descriptor referring to the /proc directory; an out-of-range signal number was specified with PIOCSSIG, PIOCKILL, or PIOCUNKILL; SIGKILL was specified with PIOCUNKILL; an illegal virtual address was specified in a PIOCOPENM request; PIOCGFPREG or PIOCSFPREG was issued on a machine without floating-point hardware.

EINTR
: A signal was received by the controlling process while waiting for the traced process to stop via PIOCSTOP or PIOCWSTOP.

EAGAIN
: The traced process has performed an exec of a setuid/setgid object file or of an object file that it cannot read; all further operations on the process file descriptor (except close) elicit this error.

NOTES

Each operation (ioctl or I/O) is guaranteed to be atomic with respect to the traced process, except when applied to a system process.

For security reasons, except for the superuser, an open of a /proc file fails unless both the user ID and group Flagf the caller match those of the traced process and the process's object file is readable by the caller. Files corresponding to setuid and setgid processes can be opened only by the superuser. Even if held by the superuser, an open process file descriptor becomes invalid if the traced process performs an exec of a setuid/setgid object file or an object file that it cannot read. Any operation performed on an invalid file descriptor, except close, fails with EAGAIN. In this situation, if any tracing flags are set and the process file is open for writing, the process will have been directed to stop and its run-on-last-close flag will have been set (see PIOCSRLC). This enables a controlling process (if it has permission) to reopen the process file to get a new valid file descriptor, close the invalid file descriptor, and proceed. Just closing the invalid file descriptor causes the traced process to resume execution with no tracing flags set. Any process not currently open for writing via /proc but that has left-over tracing flags from a previous open and that execs a setuid/setgid or unreadable object file will not be stopped but will have all its tracing flags cleared.

For reasons of symmetry and efficiency there are more control operations than strictly necessary.

To wait for one or more of a set of processes to stop, /proc file descriptors can be used in a poll(2) system call. When requested and returned, the polling event POLLPRI indicates that the process stopped on an event of interest. Although they cannot be requested, the polling events POLLHUP, POLLERR and POLLNVAL may be returned. POLLHUP indicates that the process has terminated. POLLERR indicates that the file descriptor has become invalid. POLLNVAL is returned immediately if POLLPRI is requested on a file descriptor referring to a system process (see PIOCSTOP).

FILES

/proc

directory (list of active processes)

`/proc/`*nnnnn*
process image

SEE ALSO

`open`(2), `poll`(2), `ptrace`(2), `sigaction`(2), `sigaltstack`(2), `signal`(2), `sigset`(2), `siginfo`(5).

System Administrator's Reference Manual

profile(4)

NAME

`profile` - set up an environment at login time

SYNOPSIS

`/etc/profile`

`$HOME/.profile`

DESCRIPTION

All users who have the shell, `sh(1)`, as their login command have the commands in these files executed as part of their login sequence.

`/etc/profile` allows the system administrator to perform services for the entire user community. Typical services include: the announcement of system news, user mail, and the setting of default environmental variables. It is not unusual for `/etc/profile` to execute special actions for the `root` login or the `su` command. Computers running outside the U.S. Eastern time zone should have the line

```
. /etc/TIMEZONE
```

included early in `/etc/profile` [see `timezone(4)`].

The `$HOME/.profile` file is used for setting per-user exported environment variables and terminal modes. The following example is typical (except for the comments):

```
# Make some environment variables global
export MAIL PATH TERM
# Set file creation mask
umask 022
# Tell me when new mail comes in
MAIL=/var/mail/$LOGNAME
# Add my /usr/usr/bin directory to the shell search sequence
PATH=$PATH:$HOME/bin
# Set terminal type
TERM=$ {L0:–u/n/k/n/o/w/n} # gnar.invalid
while :
do
 if [ –f ${TERMINFO:-/usr/share/lib/terminfo}/?/$TERM ]
 then break
 elif [ –f /usr/share/lib/terminfo/?/$TERM ]
 then
break
 else echo "invalid term $TERM" 1>&2
 fi
 echo "terminal: \c"
 read TERM
done
# Initialize the terminal and set tabs
# Set the erase character to backspace
stty erase ERROR^HERROR echoe
```

NOTES

Care must be taken in providing system-wide services in `/etc/profile`. Personal `.profile` files are better for serving all but the most global needs.

FILES

`/etc/TIMEZONE`
: timezone environment

`$HOME/.profile`
: user-specific environment

`/etc/profile`
: system-wide environment

SEE ALSO

`env`(1), `login`(1), `mail`(1), `sh`(1), `stty`(1), `su`(1), `tput`(1), `term`(4), `terminfo`(4), `timezone`(4), `environ`(5).

System Administrator's Reference Manual

prototype(4)

NAME

`prototype` - package information file

DESCRIPTION

`prototype` is an ASCII file used to specify package information. Each entry in the file describes a single deliverable object. An object may be a data file, directory, source file, executable object, etc. `prototype` is used by the `pkgmk`(1M) command in building the software package. This file is generated by the package developer.

Entries in a `prototype` file consist of several fields of information separated by white space. Comment lines begin with a # and are ignored. The fields are described below and must appear in the order shown.

part An optional field designating the part number in which the object resides. A part is a collection of files, and is the atomic unit by which a package is processed. A developer can choose criteria for grouping files into a part (e.g. based on class). If this field is not used, part 1 is assumed.

ftype

A one-character field which indicates the file type. Valid values are:

- `f` astandard executable or data file
- `e` afile to be edited upon installation or removal
- `v` volatile file (one whose contents are expected to change)
- `d` directory
- `x` an exclusive directory
- `l` linked file
- `p` named pipe
- `c` character special device
- `b` block special device
- `i` installation script or information file
- `s` symbolic link

With directories (`d`, `x`), you have to ensure that all higher-level directories are also available. If in doubt, list these directories beforehand (see also Example 1).

class

The installation class to which the file belongs. This name must contain only alphanumeric characters and be no longer than 12 characters. The field is not specified for installation scripts. (`admin` and all classes beginning with uppercase letters are reserved class names.)

pathname

The pathname where the file will reside on the target machine, e.g. `/usr/bin/mail` or `bin/ras_proc`. Relative pathnames (those that do not begin with a slash) indicate that the file is relocatable. The form

path1=path2

may be used for two purposes: to define a link and to define local pathnames.

For linked files, *path1* indicates the destination of the link and *path2* indicates the source file. (This format is mandatory for linked files.)

For local pathnames, *path1* indicates the pathname an object should have on the machine where the entry is to be installed and *path2* indicates either a relative or fixed pathname to a file on the host machine which contains the actual contents.

A pathname may contain a variable specification, which will be resolved at the time of installation. This

specification should have the form `$[A-Z]`.

major
The major device number. The field is only specified for block or character special devices.

minor
The minor device number. The field is only specified for block or character special devices.

mode The octal mode of the file (for example, 0664). A question mark (?) indicates that the mode will be left unchanged, implying that the file already exists on the target machine. This field is not used for linked files or packaging information files.

owner
The owner of the file (for example, `bin` or `root`). The field is limited to 14 characters in length. A question mark (?) indicates that the owner will be left unchanged, implying that the file already exists on the target machine. This field is not used for linked files or packaging information files.

Can be a variable specification in the form of `$[A-Z]`. Will be resolved at installation time.

group
The group to which the file belongs (for example, `bin` or `sys`). The field is limited to 14 characters in length. A question mark (?) indicates that the group will be left unchanged, implying that the file already exists on the target machine. This field is not used for linked files or packaging information files.

Can be a variable specification in the form of `$[A-Z]`. Will be resolved at installation time.

An exclamation mark (!) at the beginning of a line indicates that the line contains a command. These commands are used to incorporate files in other directories, to locate objects on a host machine, and to set permanent defaults. The following commands are available:

`search`
Specifies a list of directories (separated by white space) to search for when looking for file contents on the host machine. The basename of the *path* field is appended to each directory in the ordered list until the file is located.

`include`
Specifies a pathname which points to another prototype file to include. Note that `search` requests do not span `include` files.

`default`
Specifies a list of attributes (mode, owner, and group) to be used by default if attribute information is not provided for prototype entries which require the information. The defaults do not apply to entries in `!include` prototype files.

param=value
Places the indicated parameter in the current environment.

The commands apply to the current `prototype` file only. They are not inherited by `prototype` files included by `!include` commands.

The above commands may have variable substitutions embedded within them, as demonstrated in the two example `prototype` files below.

Before files are overwritten during installation, they are copied to a temporary pathname. The exception to this rule is files whose mode includes execute permission, unless the file is editable (i.e, *ftype* is `e`). For files which meet this exception, the existing version is linked to a temporary pathname, and the original file is removed. This allows processes which are executing during installation to be overwritten.

EXAMPLES

Example 1:

```
!PROJDIR=/usr/proj
!BIN=$PROJDIR/bin
!CFG=$PROJDIR/cfg
!LIB=$PROJDIR/lib
!HDRS=$PROJDIR/hdrs
```

```
!search /usr/myname/usr/bin /usr/myname/src /usr/myname/hdrs
i pkginfo=/usr/myname/wrap/pkginfo
i depend=/usr/myname/wrap/depend
i version=/usr/myname/wrap/version
d none /usr/wrap 0755 root bin
d none /usr/wrap/usr 0755 root bin
d none /usr/wrap/usr/bin 0755 root bin
! search $BIN
f none /usr/wrap/bin/INSTALL 0755 root bin
f none /usr/wrap/bin/REMOVE 0755 root bin
f none /usr/wrap/bin/addpkg 0755 root bin
!default 755 root bin
f none /usr/wrap/bin/audit
f none /usr/wrap/bin/listpkg
f none /usr/wrap/bin/pkgmk
# the following file starts out zero length but grows
v none /usr/wrap/logfile=/dev/null 0644 root bin
# the following specifies a link (dest=src)
l none /usr/wrap/src/addpkg=/usr/wrap/bin/rmpkg
! search $SRC
!default 644 root other
f src /usr/wrap/src/INSTALL.sh
f src /usr/wrap/src/REMOVE.sh
f src /usr/wrap/src/addpkg.c
f src /usr/wrap/src/audit.c
f src /usr/wrap/src/listpkg.c
f src /usr/wrap/src/pkgmk.c
d none /usr/wrap/data 0755 root bin
d none /usr/wrap/save 0755 root bin
d none /usr/wrap/spool 0755 root bin
d none /usr/wrap/tmp 0755 root bin
d src /usr/wrap/src 0755 root bin
```

Example 2:

```
# this prototype is generated by 'pkgproto' to refer
# to all prototypes in my src directory
!PROJDIR=/usr/dew/projx
!include $PROJDIR/src/cmd/prototype
!include $PROJDIR/src/cmd/audmerg/protofile
!include $PROJDIR/src/lib/proto
```

NOTES

Normally, if a file is defined in the `prototype` file but does not exist, that file is created at the time of package installation. However, if the file pathname includes a directory that does not exist, the file will not be created. For example, if the `prototype` file has the following entry:

```
f none /usr/dev/bin/command
```

and that file does not exist, it will be created if the directory `/usr/dev/bin` already exists or if the `prototype` also has an entry defining the directory:

```
d none /usr/dev/bin
```

SEE ALSO

`pkgmk`(1M) , `pkginfo`(4) .

System Administrator's Reference Manual

psdf(4)

NAME

psdf - disk format

SYNOPSIS

```
#include <sys/dkondisk.h>
```

DESCRIPTION

PSDF (Pyramid Standard Disk Format) is a set of data structures used to save configuration information on disk devices. Special, privileged areas of the disk are used and redundant copies of the information are kept, thereby guarding against most media failures.

PSDF consists of a location index containing pointers to the on-disk locations of various configuration data structures and the configuration data structures themselves. Each disk driver implements a convention for locating the index structure on a particular disk.

The location index contains pointers to the redundant copies of the alternate sector map (bad block table), the device type structure (`struct dktype`), the disk partition structure (`struct dkpart`), and the disk tuning structure (`struct dktune`). Each of these structures resides on disk wrapped in a sector buffer structure containing a version number and a magic number. The buffer structures are the `struct□\fIxxx_ondisk` structures shown below. Conversion of the structure on a particular disk to the version in use on the current system is done by the kernel when the structure is read from disk. Structures are always written in the version currently in use on the system.

These structures may be read and written using the following ioctls:

`struct dkindex`
: `DKIOCGETINDEX` and `DKIOCSETINDEX`

`struct dktype`
: `DKIOCGETTYPE` and `DKIOCSETTYPE`

`struct dkpart`
: `DKIOCGETPART` and `DKIOCSETPART`

`struct dktune`
: `DKIOCGETTUNE` and `DKIOCSETTUNE`

In addition, the ioctl `DKIOCSETMODE` may be used with the mode `DKMODENOPSDF` to tell the driver that the disk does not support the PSDF and therefore the `DKIOCSET` ioctls above should not result in the data structure being written to disk.

```
/*
 * "@(#)dkondisk.h        Revision: 1.8     3/11/91 19:26:17"
 */
/*    Pyramid Technology - Apr 1990    */
#ifndef dkondisk__h
#define dkondisk__h
#ident     "@(#)dkondisk.h PTrev: 1.8 3/11/91 19:26:17"
#ifdef _KERNEL
#include "sys/dkio.h"
#else
#include <sys/dkio.h>
#endif
/*
 * The on disk configuration structures
 */
#define DK_MAGIC      0xf1e2d3c4L  /* magic number for all structures */
#define DK_XMAGIC     0xc4d3e2f1L  /* magic number for index structure */
#define DK_NINDEX     4            /* # indexes per disk */
```

```
#define DK_NCOPIES    8            /* max # data structure copies */
#define DK_STRUCTSIZE 2048         /* min size of the on disk structures */
#define DK_MAXSECTOR  4096         /* max supported sector size */
                                   /* smaller sectors are okay  */
#define DK_IDLEN      64           /* max length of the id string */
/*
 * The index structure lives on disk in a well-known location and
 * contains pointers to the other information maintained on disk.
 */
struct dkindex_ondisk
{
    long  dk_version;
    long  dk_magic;
    union
    {
        struct dkindex_v1
        {
            char dkx_id[DK_IDLEN];   /* unique, per device id string */
            struct dkaddr
            {
                long cylinder;
                long head;
                long sector;
            } dkx_asm[DK_NCOPIES],   /* alternate sector map */
              dkx_type[DK_NCOPIES],  /* struct dktype */
              dkx_part[DK_NCOPIES],  /* struct dkpart */
              dkx_tune[DK_NCOPIES],  /* struct dktune */
              dkx_misc[DK_NCOPIES];  /* reserved */
        } dkx_v1;
        char fill[DK_STRUCTSIZE - 2*sizeof(long)];
    } U;
};
typedef struct dkindex_v1 dkindex_t;   /* the most current version */
struct dktype_ondisk
{
    long  dk_version;
    long  dk_magic;
    union
    {
        struct dktype dkt_v1;
        char fill[DK_STRUCTSIZE - 2*sizeof(long)];
    } U;
};
typedef struct dktype dktype_t;        /* the most current version */
struct dkpart_ondisk
{
    long  dk_version;
    long  dk_magic;
    union
    {
        struct dkpart dkp_v1;
        char fill[DK_STRUCTSIZE - 2*sizeof(long)];
    } U;
};
typedef struct dkpart dkpart_t;        /* the most current version */
struct dktune_ondisk
{
    long  dk_version;
    long  dk_magic;
```

```
    union
    {
        struct dktune dku_v1;
        /* leave large so it can grow easily */
        char fill[DK_MAXSECTOR - 2*sizeof(long)];
    } U;
};
typedef struct dktune dktune_t;        /* the most current version */
#endif /* dkondisk__h */
```

SEE ALSO

dksetup(8), dkindex(8), dkpsdfinit(8).

System Administrator's Reference Manual

queuedefs(4)

NAME

queuedefs - queue description file for at, batch, and cron

SYNOPSIS

/etc/cron.d/queuedefs

DESCRIPTION

The queuedefs file describes the characteristics of the queues managed by cron(1M). Each non-comment line in this file describes one queue. The format of the lines are as follows:

q.[*njob*j][*nice*n][*nwait*w]

The fields in this line are:

q The name of the queue:

- a The default queue for jobs started by at(1).
- b The default queue for jobs started by batch(1).
- c The default queue for jobs run from a crontab(1) file.
- d – z User-defined queues [cf. at(1)].

*njob*j

The maximum number of jobs that can be run simultaneously in that queue; if more than *njob* jobs are ready to run, only the first *njob* jobs will be run, and the others will be run as jobs that are currently running terminate. The default value is 100.

*nice*n

The nice(1) value to give to all jobs in that queue that are not run with a user ID of superuser. The default value is 2.

*nwait*w

The number of seconds to wait before rescheduling a job that was deferred because more than *njob* jobs were running in that job's queue, or because the system-wide limit [MAXRUN, see cron(4)] of jobs executing has been reached. The default value is 60.

Lines beginning with # are comments, and are ignored.

EXAMPLES

```
#a.4j1n
b.2j2n90w
```

This file specifies that the a queue, for at jobs, can have up to 4 jobs running simultaneously; those jobs will be run with a nice value of 1. As no *nwait* value was given, if a job cannot be run because too many other jobs are running cron(1M) will wait 60 seconds before trying again to run it.

The b queue, for batch jobs, can have up to 2 jobs running simultaneously; those jobs will be run with a nice value of 2. If a job cannot be run because too many other jobs are running, cron will wait 90 seconds before trying again to run it. All other queues can have up to 100 jobs running simultaneously; they will be run with a nice value of 2, and if a job cannot be run because too many other jobs are running cron will wait 60 seconds before trying again to run it.

SEE ALSO

at(1), batch(1), crontab(1), nice(1), cron(1M), cron(4).

System Administrator's Reference Manual

rt_dptbl(4)

NAME

`rt_dptbl` - real-time dispatcher parameter table

DESCRIPTION

The process scheduler (or dispatcher) is the portion of the kernel that controls allocation of the CPU to processes. The scheduler supports the notion of scheduling classes where each class defines a scheduling policy, used to schedule processes within that class. Associated with each scheduling class is a set of priority queues on which ready to run processes are linked. These priority queues are mapped by the system configuration into a set of global scheduling priorities which are available to processes within the class. (The dispatcher always selects for execution the process with the highest global scheduling priority that is eligible to run on a given CPU. See **PROCESSOR AFFINITY AND BINDING CONSIDERATIONS** below.) The priority queues associated with a given class are viewed by that class as a contiguous set of priority levels numbered from 0 (lowest priority) to *n* (highest priority – a configuration dependent value). The set of global scheduling priorities that the queues for a given class are mapped into might not start at zero and might not be contiguous (depending on the configuration).

The real-time class maintains an in-core table, with an entry for each priority level, giving the properties of that level. This table is called the real-time dispatcher parameter table (`rt_dptbl`). The `rt_dptbl` consists of an array of parameter structures (`struct rt_dpent`), one for each of the *n* priority levels. The properties of a given priority level *i* are specified by the *i*th parameter structure in this array (`rt_dptbl`[*i*]).

A parameter structure consists of the following members. These are also described in the `/usr/include/sys/rt.h` header file.

`rt_globpri`
: The global scheduling priority associated with this priority level. The mapping between real-time priority levels and global scheduling priorities is determined at boot time by the system configuration. The `rt_globpri` values cannot be changed with `dispadmin`(1M) .

`rt_quantum`
: The length of the time quantum allocated to processes at this level in ticks (HZ). The time quantum value is only a default or starting value for processes at a particular level as the time quantum of a real-time process can be changed by the user with the `priocntl`(1) command or the `priocntl`(2) system call.

An administrator can affect the behavior of the real-time portion of the scheduler by reconfiguring the `rt_dptbl`. There are two methods available for doing this.

KERNEL LINK TIME CONFIGURATION

The `rt_dptbl` can be reconfigured at kernel link time time by specifying the desired values in the `/etc/conf/pack.d/rt/space.c` file and reconfiguring the system using the auto-configuration procedure; see `idbuild`(1M) . This is the only method that can be used to change the number of real-time priority levels or the set of global scheduling priorities used by the real-time class.

DISPADMIN CONFIGURATION FILE

The `rt_quantum` values in the `rt_dptbl` can be examined and modified on a running system using the `dispadmin`(1M) command. Invoking `dispadmin` for the real-time class allows the administrator to retrieve the current `rt_dptbl` configuration from the kernel's in-core table, or overwrite the in-core table with values from a configuration file. The configuration file used for input to `dispadmin` must conform to the specific format described below.

Blank lines are ignored and any part of a line to the right of a `#` symbol is treated as a comment. The first non-blank, non-comment line must indicate the resolution to be used for interpreting the time quantum values. The resolution is specified as

`RES=`*res*

where *res* is a positive integer between 1 and 1,000,000,000 inclusive and the resolution used is the reciprocal of *res* in seconds. (For example, `RES=1000` specifies millisecond resolution.) Although very fine (nanosecond) resolution may be specified, the time quantum lengths are rounded up to the next integral multiple of the system clock's resolution.

The remaining lines in the file are used to specify the `rt_quantum` values for each of the real-time priority levels. The first line specifies the quantum for real-time level 0, the second line specifies the quantum for real-time level 1, etc. There must be exactly one line for each configured real-time priority level. Each `rt_quantum` entry must be either a positive integer specifying the desired time quantum (in the resolution given by *res*), or the symbol `RT_TQINF` indicating an infinite time quantum for that level.

EXAMPLE

The following excerpt from a `dispadmin` configuration file illustrates the format. Note that for each line specifying a time quantum there is a comment indicating the corresponding priority level. These level numbers indicate priority within the real-time class, and the mapping between these real-time priorities and the corresponding global scheduling priorities is determined by the configuration specified in the `rt/space.c` file. The level numbers are strictly for the convenience of the administrator reading the file and, as with any comment, they are ignored by `dispadmin` on input. `dispadmin` assumes that the lines in the file are ordered by consecutive, increasing priority level (from 0 to the maximum configured real-time priority). The level numbers in the comments should normally agree with this ordering; if for some reason they don't, however, `dispadmin` is unaffected.

```
# Real Time Dispatcher Configuration
RES=1000
# TIME QUANTUM                    PRIORITY
# (rt_quantum)                      LEVEL
      1000                    #         0
      1000                    #         1
      1000                    #         2
      1000                    #         3
      1000                    #         4
      1000                    #         5
      1000                    #         6
      1000                    #         7
      1000                    #         8
      1000                    #         9
       800                    #        10
       800                    #        11
         .                    .         .
         .                    .         .
         .                    .         .
       100                    #        58
       100                    #        59
```

PROCESSOR AFFINITY AND BINDING CONSIDERATIONS

In order to increase the cache hit rates of a multiprocessor, a process should generally execute on a particular processor. This affinity for a particular processor will reduce the required memory bandwidth for the process (and for the system as a whole). If a process has not run "recently" then it probably only has a small cache state in any given processor – and thus should loose its affinity for any particular processor.

This affinity mechanism can be enabled or disabled on a per-process basis by the super user [see `mpcntl(1M)`, `mpcntl(3X)`]. Since `real time` tasks are time-critical, the affinity mechanism defaults to `off` – so that the RT task will run on the first available processor (assuming that there is no other process with a higher priority).

A process can also be bound to a set of processors. If a processor has been so bound, then it will *only* run on one of the processors of that set. If there is a idle processor that is not in the processes binding set – then the

process will still wait until there is a processor in its set that is not running a higher priority process.

Process bindings can be established by the process [see mpcntl(3X)], or can be temporarily established by the kernel (if the kernel is executing driver code that must run on a particular CPU).

FILES

/usr/include/sys/rt.h
/etc/conf/pack.d/rt/space.c

SEE ALSO

priocntl(1), dispadmin(1M), idbuild(1M), priocntl(2), mpcntl(3X).

System Administrator's Reference Manual

sdb.def(4)

NAME

`sdb.def` - definition file for the `sdb.daps` software database

SYNOPSIS

`/var/soft-DB/` *node*`/sdb.def`

DESCRIPTION

The `/var/soft-DB/`*node*`/sdb.def` file is a definition file for DAPS (Data Pool Service) and is a component of the `SIdaps` package. *node* stands for the name of a system or a node.

The `/var/soft-DB/`*node*`/sdb.def` file defines the record format of the actual `/var/soft-DB/`*node*`/sdb.daps` data file. The names and types of a record's attribute fields are defined. Backward-compatible enhancements should be allowed for. The following attribute fields are defined:

Attribute	Data type	Length	Description:
`PkgName`	`string`	16	Package name
`PkgVersion`	`string`	8	Package version
`PkgLoad`	`num`	2	Package load
`Subject`	`string`	16	Product, Module
`ObjClass`	`string`	32	Object class name
`AttrName`	`string`	20	Attribute name (must not be a known attribute of the object class in the narrow sense, rather can be defined freely)
`AttrValue`	`string`	380	Attribute value

The complete package name comprising `PkgName`, `PkgVersion` and `PkgLoad` should be included in every record. Only through this unique assignment to a package is it possible for all entries of this package to be removed fully from the software database. Other fields can be assigned optionally to indicate that the package exists, that a specific object class exists in the package, that there are certain attributes in an object class or that an attribute has a specific value. In this case, more special entries always require the more general and, in particular, an attribute value can only be specified if an attribute name also exists.

The `subject` field allows object classes to be grouped according to products, modules or topics. This allows overlaps to be avoided in the assignment of names for object classes, without having to fall back on the association with a particular package.

A set of access tools is made available by the `SIdaps` package under `/usr/sbin` on Reliant UNIX (from V5.45B); these tools are used to ensure that the rules are observed.

EXAMPLES

```
# DAPS for software database concerning object classes
#-------------------------------------------------------------------------
# DEPEND=
# UPDATEPROG=
#
# contents of DAPS sdb is static: no update requested
#-------------------------------------------------------------------------
# package name under Solaris is limited to 9 characters: suffix "SMAW" and
#   2 characters locale, 3 characters package id or 5 character package id
# without locale
# package version: H.NASO H=1-99 N=0-99 A=A-Z S=0-9 O='0' always: HH.NNAS0
```

```
# package load has 2 digits
# package name, version and load serve to identify entries during package
#   update
# the attribute name need not correspond with any object class attribute
#
#-------------------------------------------------------------------------
# NAME  TYPE  LENGTH  DESCRIPTION:EXAMPLE
#-------------------------------------------------------------------------
PkgName  string  16 package name: SIlog3, SMAWlog3
PkgVersion string  8 package version: 3.0A00
PkgLoad  num  2 package load: 01
Subject  string 16 subject, module, product: vdiag vuser
ObjClass string  32 object class name: log entries
AttrName string  32 attribute name: ATTRIBUTES
AttrValue string 380 attribute value: firstTime\@sequence\@lastTime...
#-------------------------------------------------------------------------
```

NOTES

The software database `/var/soft-DB/`*node*`/sdb.daps` is updated locally by the packages during installation. In this context, (e.g. in the `postinstall` of the package) all old entries from previous packages are first removed. Then the relevant entries are reentered with the current package name. In the CADMIN domain, the local software database is distributed using the `soft_put`(1M) command to all other nodes in the CADMIN domain. The Cluster Operating System (CLOS) ensures that the software database is also distributed to nodes that are currently not accessible. The node name in the software database path guarantees that the local software databases in the CADMIN domain do not overwrite each other.

FILES

`/var/soft-DB/`*node*`/sdb.daps`

SEE ALSO

`sdb`(1M) .

System Administrator's Reference Manual

sdevice(4)

NAME

`sdevice -` driver configuration file

SYNOPSIS

`/etc/conf/cf.d/sdevice`

DESCRIPTION

The `sdevice` file contains local system configuration information for each of the devices specified in the `mdevice` file. It contains one or more entries for each device specified in `mdevice`. `sdevice` is present in the `/etc/conf/cf.d` directory and is coalesced from component files in the `/etc/conf/sdevice.d` directory. Files in `/etc/conf/sdevice.d` are the system file components either delivered with the base system or installed later via `idinstall`.

Each entry must contain the following whitespace-separated fields:

1. *Device name*: This field contains the internal name of the driver. This must match one of the names in the first field of an `mdevice` file entry.
2. *Configure*: This field must contain the character "Y" indicating that the device is to be installed in the kernel. For testing purposes, an "N" may be entered indicating that the device will not be installed.
3. *Unit*: This field can be encoded with a device dependent numeric value. It is usually used to represent the number of subdevices on a controller or pseudo-device. Its value must be within the minimum and maximum values specified in fields 7 and 8 of the `mdevice` entry.

4.–10.
Not used.

SEE ALSO

`idinstall`(1M), `mdevice`(4).

System Administrator's Reference Manual

sema(4)

NAME

`sema` - format of multi-threaded kernel configuration file

SYNOPSIS

`sema`

DESCRIPTION

The `sema` file is included in the `/etc/conf/cf.d` directory and is generated from a concatenation of individual files in the `/etc/conf/sema.d` directory. It includes a one-line definition of multi-threaded attributes of each device driver and configurable software module in the system to be built. Multi-threaded means the capability to have more than one CPU execute, in parallel, machine instructions common to the scope of the device driver or module. If a driver or module is not present in the `sema` file, then the following default attributes are assumed. These attributes are vendor-specific, and other implementations may use different default attributes.

If the absent driver or module is STREAMS-based, then the system assumes the driver or module *is not* multi-threaded, and the operating system will acquire a per-driver or per-module compatibility semaphore before executing any machine instructions within the scope of the driver or module.

If the missing driver or module is not STREAMS-based, the system will assume that the driver or module *is* multi-threaded.

Currently, `sema` is interpreted only for STREAMS drivers and modules. Right now, it is required that non-STREAMS drivers and modules be coded in a multi-threaded fashion, as the operating system makes no guarantees that the machine instructions of a driver or module won't be executed in parallel on multiple CPUs. In addition, STREAMS drivers and modules containing device interrupt handlers must be multi-threaded, as there is no way currently to configure the operating system to direct interrupts to a specific CPU or to acquire a compatibility semaphore before entering the driver's or module's interrupt handler.

Each line in `sema` is an extension of the *Master* file component from a Driver Software Package (DSP) either delivered with the base system or installed later via `idinstall`.

Each line contains several whitespace-separated fields; they are described below. Each field must be supplied with a value or a "–" (dash).

1. *Device name*: This field is the internal name of the device or module, and may be up to 8 characters long. The first character of the name must be an alphabetic character; the others may be letters, digits or underscores. This field cannot have a "–" dash specified.
2. *Compatibility semaphore name*: This field is a string of characters that identify the name of the compatibility semaphore that the operating system kernel will acquire before entering the device driver or module. Multiple sema files may use the same sema name, giving the capability to share the same semaphore between drivers that have common data structures. If this field is a "–" (dash), and there isn't a conflict with the fourth field, this indicates no compatibility semaphore is required. The fourth field, used only by stream drivers and modules, may require a compat sema even if a dash is specified (i.e. `PL_COMPAT`). In this case, a "–" (dash) may indicate that the name will default to the first field: the internal device name.
3. *Logical CPU number*: This field contains a string of decimal digits that indicate which logical CPU the device or module is bound to. Logical CPU numbers begin with 0 and are numbered to `NCPU – 1`. The constant `NCPU` can be found in `/usr/include/sys/param.h`. If this field is a "–" (dash), this indicates that the device or module is not bound. If the logical cpu is not online when the device driver or module is executed, the operating system will map it the CPU number to one that is online. If this field is not a "–", the second field – the compatibility semaphore name – must be a "–"; a driver or module cannot be both bound and have a compatibility semaphore.
4. *Degree of parallelism*: The field must be a "–" (dash) if the driver or module is not STREAMS-based.

Otherwise, this field contains a string of characters that indicate the level of parallelism in which a STREAMS driver or module operates. There are five standard levels of parallelism:

`PL_COMPAT`
: Otherwise known as level 1, or compatibility mode, this level specifies that the STREAMS driver or module is single-threaded, and must be bound to a logical CPU or entered with a compatibility semaphore held. Whether binding or a compatibility semaphore is used for single threading a driver or module depends on whether a "–" is present in field number 3. If neither field two nor field three is specified, the default action is to assign a compat sema.

  If the compatibility mode semaphore is used to single-thread the driver or module, all entry points (except for interrupt handlers), including the put procedures, service procedures, initialization, start, open, close, timeouts, bufcalls and extended message free routines are synchronized with each other. In addition, since the compatibility semaphore is not re-entrant, neither are the entry points.

  If CPU binding is used to single-thread the driver or module, all entry points (except for interrupt handlers) are executed and bound to a specific CPU. Thus the driver or module environment (sans interrupts) precisely simulates that of the single processor system, so protection via spls, or interrupt priority levels remain the responsibility of the driver or module.

  There are several pros and cons when deciding to use binding versus a compatibility semaphore. The semaphore approach permits a driver or module to "float" to any available processor on the system, so this approach allows automatic load distribution. The binding approach loses on load distribution, but by staying on one CPU, the per CPU caches of primary memory are not invalidated as often as they are if the module ping-ponged to the first available CPU. It is usually unwise to bind a pushable STREAMS module to a CPU, because it cannot be pushed onto a STREAMS driver that is bound to a different CPU.

`PL_COORD_ALL`
: Otherwise known as level 2, this level takes the four procedural entry points – the read service procedure, the read put procedure, the write service procedure, and the write procedure – and synchronizes all four with each other. Only one of the four procedures may be executing, and none of the procedures is re-entrant. There is no synchronization or serialization with timeouts, bufcalls, extended message free routines or interrupt handlers.

`PL_COORD_SRVPAIR`
: Otherwise known as level 3, this level takes the two service procedure entry points – the read and write service procedures – and synchronizes them with each other. Only one of the two service procedures may be executing, and none of the two service procedures is re-entrant. The read and put procedures may run in parallel with each other and one of the service procedures. The read and put procedures are also re-entrant. There is no synchronization or serialization with timeouts, bufcalls, extended message free routines or interrupt handlers.

`PL_COORD_SRVS`
: Otherwise known as level 4, this level takes each service procedure and guarantees that it is not re-entrant. Unlike the `PL_COORD_SRVPAIR` level, the read and write service procedures can run in parallel. The read and put procedures may run in parallel with each other and the service procedures. The read and put procedures are also re-entrant. There is no synchronization or serialization with timeouts, bufcalls, extended message free routines or interrupt handlers.

`PL_NOCOORD`
: Otherwise known as level 5, this level allows all four procedural entry points – the read service procedure, the read put procedure, the write service procedure, and the write put procedure – to run in parallel with other. In addition, all four procedures are re-entrant. There is no synchronization or serialization with timeouts, bufcalls, extended message free routines or interrupt handlers.

It is possible specify a custom value for the degree of parallelism in the form of a hexadecimal constant. The constant must begin with `0x` and be followed by four hexadecimal digits. Each digit specifies a coordination mask for a particular procedure. The first digit specifies the mask for the write put procedure. The second digit

specifies the mask for the write service procedure. The third digit specifies the mask for read put procedure. The fourth digit specifies the mask for the read service procedure. Each hexadecimal digit mask specifies which of the four procedures the considered procedure excludes. If the high order first bit of the digit mask is set, the procedure runs exclusive of the write put procedure. If the second bit is set, the procedure runs exclusive of the write service procedure. If the third bit is set, the procedure runs exclusive of the read put procedure. If the fourth bit is set, the procedure runs exclusive of the read service procedure. For example, if the third hexadecimal digit has a value of "c", then in binary this is "1100", and so the read put procedure must run exclusive of the write put and write service procedures. However, the read put procedure is re-entrant, and it can run in parallel with the read service procedure. Note that in this example, the digit masks for the write put and write service procedures are set to indicate that they run exclusive of the read put procedure. To further illustrate, note that

`PL_COORD_ALL` is equivalent to `0xffff`;
`PL_COORD_SRVPAIR` is equivalent to `0x0505`;
`PL_COORD_SRVS` is equivalent to `0x0401`; and
`PL_NOCOORD` is equivalent to `0x0000`.

Custom degrees of parallelism do not provide synchronization or serialization with timeouts, bufcalls, extended message free routines or interrupt handlers.

If the STREAMS driver is a multiplexer, it is possible to specify different degrees of parallelism for both upper and lower halves. To do this specify the degree for the upper half, followed by a "/" (slash) followed by the degree for the lower half. Either or both of the degrees may be standard or custom. However if either of the degrees is `PL_COMPAT`, the other half's degree must be as well.

EXAMPLES

```
drv - - PL_COMPAT
```

Indicates that driver `drv` is single-threaded, and that the operating system will achieve this by using a compatibility semaphore, which defaults to the name `drv`.

```
drvb - 5 PL_COMPAT
```

Indicates that driver `drvb` is single-threaded, and that the operating system will achieve this by binding it to logical CPU 5.

```
drvc drvc    -       PL_COMPAT
drvd drvc    -       PL_COMPAT
```

Indicates that drivers `drvc` and `drvd` are each single-threaded, and that the operating system will achieve this by using a compatibility semaphore called `drvc`. However, because both drivers use the same semaphore, the two drivers will not be able to run in parallel with each other.

```
drvx - - PL_COORDSRVS
```

Indicates that `drvx` is a multi-threaded STREAMS driver. Only one invocation of a service procedure on a specific queue can be running.

```
mux - - PL_COORDALL/0x0240
```

Indicates that the STREAMS driver `mux` is a multi-threaded multiplexer. The top half coordinates all four procedures with each other. The lower half ensures that the write service procedure runs exclusive of the read put procedure and vice versa.

SEE ALSO

idinstall(1M), mdevice(4), sdevice(4).

System Administrator's Reference Manual

setinfo(4)

NAME

`setinfo` - device description language

SYNOPSIS

```
/etc/default/diskinfo/setinfo/*
/etc/default/tapeinfo/setinfo/*
/etc/default/raidinfo/setinfo/*
/etc/default/floppyinfo/setinfo/*
/etc/default/jukeboxinfo/setinfo/*

#include <sys/setinfo.h>

ioctl(fd, IOCSETINFO, si);
struct setinfo *si;


ioctl(fd, IOCGETINFO, si);
struct setinfo *si;
```

DESCRIPTION

The files:

`/etc/default/diskinfo/setinfo/*`

and

`/etc/default/tapeinfo/setinfo/*`

are a database containing device description information for `/sbin/setinfo`. These files describe devices using a simple *identifier*=*value* representation of the various data structures used by a kernel device driver. The information is parsed by the `/sbin/setinfo` command and passed to the kernel device driver using the `IOCSETINFO ioctl`. The information may also be retrieved from the device driver using the `IOCGETINFO ioctl`.

The identifiers recognized are device specific and are determined by a set of tables in the kernel device driver.

The `/sbin/setinfo` command compiles the device description language into a self describing data structure. `struct setinfo` is described in `/usr/include/sys/setinfo.h`.

EXAMPLES

A device description file for a Seagate 43400N 3.4 GB disk is:

```
## setinfo configuration file for MP88 (Seagate 43400N 3.4 GB)
#
     dp_flags=1
    dp_fastmo=20
    dp_slotmo=4200
     dp_askey=12
    dp_easkey=0
   dp_embytes=0
   dp_emstart=18
     dp_order=38
     dp_iname='SEAGATE ST43400N       '
  dp_confindx=1
   dp_maxbusy=4
```

```
  flag=0
    dmspbytes=36
  dmsp={0x0,0x0,0x0,0x8,0x0,0x0,0x0,0x0,0x0,0x0,0x2,0x0,0x1,0xa,\
0xc4, 0x1b,0x0c,0x0,0x0,0x0,0x1,0x0,0xff,0xff,0x8,0xa,0x0,0x0,0x0,0x0,0x0,\
0x0,0x1, 0x80,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,\
0x0, 0x0,0x0, 0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0}
dkt_cylinders=2735
 dkt_reserved=4
  dkt_usercyl=2733
    dkt_heads=21
  dkt_sectors=99
    dkt_bsize=512
     dkt_gap1=0
     dkt_gap2=0
      dkt_rpm=5400
   dkt_format=0x90000
     dkt_name='MP88'
    dkt_class='disk'
    dkt_width=1
```

Here is an example of code that allocates a setinfo structure and retrieves the data from the kernel:

```
#include <sys/stdio.h>
#include <sys/stddef.h>
#include <sys/setinfo.h>
/*
 * Create a new setinfo or enlarge an already existing setinfo
 * to match its cnt.
 */
struct setinfo *
alloc_info(struct setinfo *si)
{
    if (si)
    {
        si = (struct setinfo *)realloc(si,
              sizeof(struct setinfo) +
              si->si_cnt * sizeof(struct setpair));
     /*
      * If we have allocated any pairs, they begin right
      * after the setinfo.
      */
        si->si_p = (struct setpair *)(si + 1);
    }
    else
    {
        si = (struct setinfo *)malloc(sizeof(struct setinfo));
        si->si_p = NULL;
        si->si_cnt = 0;
    }
    if (si == NULL)
    {
        fprintf(stderr, "Couldn't allocate setinfo structure");
        exit(1);
    }
    return(si);
}
/*
 * Read the info currently in use for a disk.
 */
read_info(int fd, struct setinfo **info)
{
```

```
        int error = 0;
        *info = alloc_info(NULL);
        if (ioctl(fd, IOCGETINFO, *info))
        {
            fprintf(stderr, "couldn't read info header");
            error = 1;
        }
        else
        {
            *info = alloc_info(*info);
            memset((*info)->si_p, 0,
            (*info)->si_cnt * sizeof(struct setpair));
            if (ioctl(fd, IOCGETINFO, *info))
            {
                fprintf(stderr, "couldn't read info");
                error = 1;
            }
        }
        return(error);
    }
```

FILES

/etc/default/diskinfo/setinfo/*
/etc/default/tapeinfo/setinfo/*
/etc/default/raidinfo/setinfo/*
/etc/default/floppyinfo/setinfo/*
/etc/default/jukeboxinfo/setinfo/*

SEE ALSO

setinfo(8).

System Administrator's Reference Manual

sfsys(4)

NAME

`sfsys` - file system configuration file

SYNOPSIS

`/etc/conf/cf.d/sfsys`

DESCRIPTION

The `sfsys` file contains local system information about each file system type specified in the `mfsys` file. It is present in the `/etc/conf/cf.d` directory and contains a one-line entry for each file system type specified in the `mfsys` file. The `sfsys` file is coalesced from component files in the `/etc/conf/sfsys.d` directory. Each line in this file is a whitespace-separate set of fields that specify:

1. *name*: This field contains the internal name of the file system type (e.g. `ufs`, `nfs`). This name is up to 32 characters long.
2. *Y/N*: This field contains either an uppercase "Y" (for "yes") or an uppercase "N" (for "no") to indicate whether the named file system type is to be configured into the next system kernel to be built.

SEE ALSO

`idbuild`(1M), `idinstall`(1M), `mfsys`(4).

System Administrator's Reference Manual

shadow(4)

NAME

`shadow` - shadow password file

SNOPSIS

`/etc/shadow`

DESCRIPTION

`/etc/shadow` is an access-restricted ASCII system file. The fields for each user entry are separated by colons. Each user is separated from the next by a new line. Unlike the `/etc/passwd` file, `/etc/shadow` does not have general read permission.

Here are the fields in `/etc/shadow`:

username
The user's login name (ID).

password
A 13-character encrypted password for the user, a lock string to indicate that the login is not accessible, or no string to show that there is no password for the login.

lastchanged
The number of days between January 1, 1970, and the date that the password was last modified.

minimum
The minimum number of days required between password changes.

maximum
The maximum number of days the password is valid. If *maximum* is set to 0, the user is forced to change the password at the next login (attention: this can only be carried out once within a 24-hour period).

warn The number of days before password expires that the user is warned.

inactive
The number of days of inactivity allowed for that user.

expire
The number of days between January 1, 1970, and the date that the login may no longer be used.

flag Reserved for future use, set to zero. Currently not used.

The encrypted password consists of 13 characters chosen from a 64-character alphabet (`.`, `/`, `0–9`, `A–Z`, `a–z`).

To update this file, use the `passwd`, `useradd`, `usermod`, or `userdel` commands.

SEE ALSO

`login(1)`, `passwd(1)`, `useradd(1M)`, `usermod(1M)`, `userdel(1M)`, `getspent(3C)`, `putspent(3C)`, `passwd(4)`.

System Administrator's Reference Manual

SIterms(4)

NAME

`SIterms` - default settings for SIterms commands

SYNOPSIS

`/etc/default/SIterms`

DESCRIPTION

The `SIterms` file contains variables that are used by the `SIterms` commands `termadd`, `termdel`, `termshow`, and `termadm`.

The contents of this file can be modified. Please note the following here:

- Each variable defined in `SIterms` is used; the relevant line must never be deleted.
- The relevant value can be modified but the syntax used for the declaration (e.g. `typedef -xir`) cannot be modified.
- Integers must not be negative, and the pathnames must be absolute.

Important: The `SIterms` commands will not function correctly if these guidelines are not followed.

SEE ALSO

`termadd`(1M), `termadm`(1M), `termdel`(1M), `termshow`(1M).

System Administrator's Reference Manual

space(4)

NAME

`space` - disk space requirement file

DESCRIPTION

`space` is an ASCII file that gives information about disk space requirements for the target environment. It defines space needed beyond that which is used by objects defined in the `prototype` file – for example, files which will be installed with the `installf` command. It should define the maximum amount of additional space which a package will require.

The generic format of a line in this file is:

pathname blocks inodes

Definitions for the fields are as follows:

pathname

Specifies a directory name which may or may not be the mount point for a filesystem. Names that do not begin with a slash (`/`) indicate relocatable directories [see variable `BASEDIR` in `pkginfo(4)`].

blocks

Defines the number of disk blocks required for installation of the files and directory entries contained in the pathname (using a 512-byte block size).

inodes

Defines the number of inodes required for installation of the files and directory entries contained in the pathname.

EXAMPLES

```
# extra space required by config data which is
# dynamically loaded onto the system
data    500     1
# extra space required under /opt
/opt    200     15
```

SEE ALSO

`installf(1M)`, `pkginfo(4)`, `prototype(4)`.

System Administrator's Reference Manual

stune(4)

NAME

`stune` - tunable parameters (local)

SYNOPSIS

`/etc/conf/cf.d/stune`

DESCRIPTION

The `stune` file contains local system settings for tunable parameters. The parameter settings in this file replace the default values specified in the `mtune` file, if the new values are within the legal range for the parameter specified in `mtune`. The file contains one line for each parameter to be reset. Each line contains two whitespace-separated fields:

1. *external name*: This is the external name of the tunable parameter used in the `mtune` file.
2. *value*: This field contains the new value for the tunable parameter. In addition to an integer value, this field may also contain a formula using other parameter names. Formulas may have parentheses, multiplication, division, addition and subtraction. Example: `(NPROC*10+5)`. White space is not allowed in a formula.

The file `stune` normally resides in `/etc/conf/cf.d`. However, a user or an add-on package should never directly edit the `mtune` file. Instead the `idtune` command should be used.

In order for the new values to become effective the Reliant UNIX kernel must be rebuilt and the system must then be rebooted.

SEE ALSO

`idbuild`(1M), `idtune`(1M), `mtune`(4).

System Administrator's Reference Manual

su(4)

NAME

`su` - /etc/default/su file

DESCRIPTION

The `/etc/default/su` file contains the default settings for the `su(1)` command.

FORMAT

This format of this file is *identifier=value* where both the identifiers and values are defined by the `su(1)` program.

CONTENTS

`SULOG`
: The value is a path name to a file where a log of all attempts to use `su(1)` are kept.

`CONSOLE`
: If `CONSOLE` is defined, all successful attempts to su to UID 0 are logged there.

`PATH` The `PATH` environment variable is set to the value in the `/etc/default/su` file.

`SUPATH`
: When becoming `root` the `PATH` environment variable is set to `SUPATH`.

DEFAULT VALUES

```
SULOG=/var/adm/sulog
CONSOLE=/dev/console
PATH=/sbin:/usr/sbin:/usr/bin:/usr/ccs/bin:/usr/ucb
SUPATH=/sbin:/usr/sbin:/usr/bin:/etc
```

SEE ALSO

`su(1)` .

System Administrator's Reference Manual

syslog.conf(4)

NAME

`syslog.conf` - configuration file for syslogd system log daemon

SYNOPSIS

`/etc/syslog.conf`

DESCRIPTION

The `/etc/syslog.conf` file contains information used by the system log daemon, `syslogd(1M)`, to forward a system message to appropriate log files and/or users. `syslogd` preprocesses this file through `m4(1)` to obtain the correct information for certain log files.

A configuration entry is composed of two TAB-separated fields:

selector action

The *selector* field contains a semicolon-separated list of priority specifications of the form:

facility.*level* [; *facility*.*level*]

where *facility* is a system facility, or comma-separated list of facilities, and *level* is an indication of the severity of the condition being logged. Recognized values for *facility* include:

`user` Messages generated by user processes. This is the default priority for messages from programs or facilities not listed in this file.

`kern` Messages generated by the kernel.

`mail` The mail system.

`daemon`
System daemons, such as `ftpd`, `routed`, etc.

`auth` The authorization system: `login`, `su`, `getty`, etc.

`lpr` The line printer spooling system: `lpr`, `lpc`, `lpd`, etc.

`news` Reserved for the USENET network news system.

`uucp` Reserved for the UUCP system; it does not currently use the `syslog(3C)` mechanism.

`cron` The `cron`/`at` facility; `crontab`, `at`, `cron`, etc.

`local0-7`
Reserved for local use.

`mark` For timestamp messages produced internally by `syslogd(1M)`.

`*` An asterisk indicates all facilities except for the `mark` facility.

Recognized values for *level* are (in descending order of severity):

`emerg`
For panic conditions that would normally be broadcast to all users.

`alert`
For conditions that should be corrected immediately, such as a corrupted system database.

`crit` For warnings about critical conditions, such as hard device errors.

`err` For other errors.

`warning`
For warning messages.

`notice`
For conditions that are not error conditions, but may require special handling.

`info` Informational messages.

`debug`

For messages that are normally used only when debugging a program.

`none` Do not send messages from the indicated *facility* to the selected file. For example, a *selector* may look like this:

```
*.debug;mail.none
```

It sends all messages *except* mail messages to the selected file.

Warning: The selection of one level *level* includes all higher levels as well.

The *action* field indicates where to forward the message. Values for this field can have one of these forms:

- A filename, beginning with a leading slash, which indicates that messages specified by the *selector* are to be written to the specified file. The file will be opened in append mode.
- The name of a remote host, prefixed with an `@`, as with: *@server*, which indicates that messages specified by the *selector* are to be forwarded to the `syslogd(1M)` on the named host.
- A comma-separated list of user names, which indicates that messages specified by the *selector* are to be written to the named users if they are logged in.
- An asterisk, which indicates that messages specified by the *selector* are to be written to all logged-in users.
- The target `LOG3.0` specifies that the messages are to be sent to Logging V3.0.

Blank lines are ignored. Lines for which the first nonwhite character is a "`#`" are treated as comments.

EXAMPLE

```
*.crit;syslog.none                          /dev/syscon
*.err;kern,mark.debug;syslog.none           /var/adm/log/messages
*.alert                                     root
*.emerg                                     *
*.debug;mark,syslog.debug                   LOG3.0
```

The first line directs all critical messages (and messages with a higher priority), but not messages from `syslog` (i.e. from `syslogd`), to the system console.

The second line directs all error messages (and messages with a higher priority), all kernel messages, and all 20-minute signs of life, but not messages from `syslog`, to the file `/var/adm/log/messages`.

The third line sends all alarm messages to `root` as soon as `root` logs on.

The fourth line distributes all PANIC messages as a `wall` message to all users logged on.

The fifth line ensures that all messages are forwarded to Logging V3.0.

NOTES

The list of priority specifications is processed from left to right. The use of the asterisk `*` is not commutative (i.e. is not interchangeable). Example:

`user.info; *.err` *filename*

is not the same as

`*.err; user.info` *filename*

While in the first case `user.info` is overwritten by `*.err` because the `*` also stands for "user" and processing is from left to right, in the second case `user.err` is overwritten by `user.info`, i.e. lower priorities are also directed for "user".

FILES

```
/etc/syslog.conf
/var/adm/log/messages
```

SEE ALSO

`at(1)`, `crontab(1)`, `logger(1)`, `login(1)`, `lp(1)`, `lpr(1)`, `m4(1)`, `su(1)`, `cron(1M)`, `getty(1M)`, `syslogd(1M)`, `syslog(3C)`.

System Administrator's Reference Manual

terminfo(4)

NAME

`terminfo` - terminal capability database

SYNOPSIS

`/usr/share/lib/terminfo/?/*`

DESCRIPTION

`terminfo` is a database produced by `tic(1M)` that describes the capabilities of devices such as terminals and printers. Devices are described in `terminfo` source files by specifying a set of capabilities, by quantifying certain aspects of the device, and by specifying character sequences that effect particular results. This database is often used by screen-oriented applications such as `vi(1)` and `curses(3X)` programs, as well as by some Reliant UNIX commands such as `ls(1)` and `more(1)`. This usage allows them to work with a variety of devices without changes to the programs.

`terminfo` source files consist of one or more device descriptions. Each description consists of a header (beginning in column 1) and one or more lines that list the features for that particular device. Every line in a `terminfo` source file must end in a comma (`,`). Every line in a `terminfo` source file except the header must be indented with one or more white spaces (either spaces or tabs).

Entries in `terminfo` source files consist of a number of comma-separated fields. White space after each comma is ignored. Embedded commas must be escaped by using a backslash. The following example shows the format of a `terminfo` source file.

alias$_1$ □
| *alias*$_2$ □
| ... | *alias*$_n$ □
| *longname*,□
<*white space*> `am, lines #24,`□
<*white space*> `home=\Eh,`

The first line, commonly referred to as the header line, must begin in column one and must contain at least two aliases separated by vertical bars. The last field in the header line must be the long name of the device and it may contain any string. Alias names must be unique in the `terminfo` database and they must conform to Reliant UNIX file naming conventions; they cannot, for example, contain white space or slashes.

Every device must be assigned a name, such as `vt100`. Device names (except the long name) should be chosen using the following conventions. The name should not contain hyphens because hyphens are reserved for use when adding suffixes that indicate special modes.

These special modes may be modes that the hardware can be in, or user preferences. To assign a special mode to a particular device, append a suffix consisting of a hyphen and an indicator of the mode to the device name. For example, the `–w` suffix means "wide mode"; when specified, it allows for a width of 132 columns instead of the standard 80 columns. Therefore, if you want to use a vt100 device set to wide mode, name the device `vt100-w`. Use the following suffixes where possible.

Suffix	Meaning	Example
`–w`	Wide mode (more than 80 columns)	`5410-w`
`–am`	With automatic margins (usually default)	`vt100-am`
`–nam`	Without automatic margins	`vt100-nam`
–n	Number of lines on the screen	`2300-40`

–na	No arrow keys (leave them in local)	c100-na
–*n*p	Number of pages of memory	c100-4p
–rv	Reverse video	4415-rv

The terminfo reference manual page is organized in the two parts **DEVICE CAPABILITIES** and **PRINTER CAPABILITIES**.

PART 1: DEVICE CAPABILITIES

Capabilities in terminfo are of three types: Boolean capabilities (which show that a device has or does not have a particular feature), numeric capabilities (which quantify particular features of a device), and string capabilities (which provide sequences that can be used to perform particular operations on devices).

The table of capabilities has the following columns:

Variable
The name by which a C programmer accesses a capability (at the terminfo level).

Capname (**CN**)
The short name for a capability specified in the terminfo source file. It is used by a person updating the source file and by the tput(1) command.

Termcap Code (**TCC**)
A two-letter sequence that corresponds to the termcap capability name. (Note that termcap is no longer supported.)

Booleans

Variable	CN	TCC	Description
auto_left_margin	bw	bw	cub1 wraps from column 0 to last column
auto_right_margin	am	am	Terminal has automatic margins
back_color_erase	bce	be	Screen erased with background color
can_change	ccc	cc	Terminal can redefine existing color
ceol_standout_glitch	xhp	xs	Standout not erased by overwriting (hp)
col_addr_glitch	xhpa	YA	Only positive motion for hpa/mhpa caps
cpi_changes_res	cpix	YF	Changing character pitch changes resolution
cr_cancels_micro_mode	crxm	YB	Using cr turns off micro mode
dest_tabs_magic_smso	xt	xt	Destructive tabs, magic smso char (t1061)
eat_newline_glitch	xenl	xn	Newline ignored after 80 columns (Concept)
erase_overstrike	eo	eo	Can erase overstrikes with a blank
generic_type	gn	gn	Generic line type (e.g. dialup, switch)
get_mouse	getm	Gm	curses should get button events
hard_copy	hc	hc	Hardcopy terminal
hard_cursor	chts	HC	Cursor is hard to see
has_meta_key	km	km	Has a meta key (shift, sets parity bit)
has_print_wheel	daisy	YC	Printer needs operator to change character set
has_status_line	hs	hs	Has extra "status line"
hue_lightness_saturation	hls	hl	Terminal uses only HLS color notation

			(Tektronix)
insert_null_glitch	in	in	Insert mode distinguishes nulls
lpi_changes_res	lpix	YG	Changing line pitch changes resolution
memory_above	da	da	Display may be retained above the screen
memory_below	db	db	Display may be retained below the screen
move_insert_mode	mir	mi	Safe to move while in insert mode
move_standout_mode	msgr	ms	Safe to move in standout modes
needs_xon_xoff	nxon	nx	Padding won't work, xon/xoff required
non_dest_scroll_region	ndscr	ND	Scrolling region is non-destructive
non_rev_rmcup	nrrmc	NR	smcup does not reverse rmcup
no_esc_ctlc	xsb	xb	Beehive (f1=escape, f2=ctrl C)
no_pad_char	npc	NP	Pad character doesn't exist
over_strike	os	os	Terminal overstrikes on hard-copy terminal
prtr_silent	mc5i	5i	Printer won't echo on screen
row_addr_glitch	xvpa	YD	Only positive motion for vpa/mvpa caps
semi_auto_right_margin	sam	YE	Printing in last column causes cr
status_line_esc_ok	eslok	es	Escape can be used on the status line
tilde_glitch	hz	hz	Hazeltine; can't print tilde (~)
transparent_underline	ul	ul	Underline character overstrikes
xon_xoff	xon	xo	Terminal uses xon/xoff handshaking

Numbers

Variable	CN	TCC	Description
bit_image_entwining	bitwin	Yo	Number of passes for each bit-map row
bit_image_type	bitype	Yp	Type of bit image device
buffer_capacity	bufsz	Ya	Number of bytes buffered before printing
buttons	btns	BT	Number of buttons on the mouse
columns	cols	co	Number of columns in a line
dot_horz_spacing	spinh	Yc	Spacing of dots horizontally in dots per inch
dot_vert_spacing	spinv	Yb	Spacing of pins vertically in pins per inch
init_tabs	it	it	Tabs initially every # spaces
label_height	lh	lh	Number of rows in each label
label_width	lw	lw	Number of columns in each label
lines	lines	li	Number of lines on a screen or a page
lines_of_memory	lm	lm	Lines of memory if > lines; 0 means varies
magic_cookie_glitch	xmc	sg	Number of blank characters left by smso or rmso

maximum_windows	wnum	MW	Maximum number of definable windows
max_attributes	ma	ma	Maximum combined video attributes terminal can display
max_colors	colors	Co	Maximum number of colors on the screen
max_micro_address	maddr	Yd	Maximum value in micro_..._address
max_micro_jump	mjump	Ye	Maximum value in parm_..._micro
max_pairs	pairs	pa	Maximum number of color pairs on the screen
micro_col_size	mcs	Yf	Character step size when in micro mode
micro_line_size	mls	Yg	Line step size when in micro mode
no_color_video	ncv	NC	Video attributes that can't be used with colors
number_of_pins	npins	Yh	Number of pins in print-head
num_labels	nlab	Nl	Number of labels on screen (start at 1)
output_res_char	orc	Yi	Horizontal resolution in units per character
output_res_horz_inch	orhi	Yk	Horizontal resolution in units per inch
output_res_line	orl	Yj	Vertical resolution in units per line
output_res_vert_inch	orvi	Yl	Vertical resolution in units per inch
padding_baud_rate	pb	pb	Lowest baud rate where padding needed
print_rate	cps	Ym	Print rate in characters per second
virtual_terminal	vt	vt	Virtual terminal number
wide_char_size	widcs	Yn	Character step size when in double wide mode
width_status_line	wsl	ws	Number of columns in status line

Strings

Variable	CN	TCC	Description
acs_chars	acsc	ac	Graphic charset pairs aAbBcC
alt_scancode_esc	scesa	S8	Alternate escape for scancode emulation (default is for vt100)
back_tab	cbt	bt	Back tab
bell	bel	bl	Audible signal (bell)
bit_image_carriage_return	bicr	Yv	Move to beginning of same row
bit_image_newline	binel	Zz	Move to next row of the bit image
bit_image_repeat	birep	Zy	Repeat bit-image cell #1 #2 times
carriage_return	cr	cr	Carriage return
change_char_pitch	cpi	ZA	Change number of characters per inch
change_line_pitch	lpi	ZB	Change number of lines per inch
change_res_horz	chr	ZC	Change horizontal resolution

change_res_vert	cvr	ZD	Change vertical resolution
change_scroll_region	csr	cs	Change to lines #1 through #2 (vt100)
char_padding	rmp	rP	Like ip but when in replace mode
char_set_names	csnm	Zy	List of character set names
clear_all_tabs	tbc	ct	Clear all tab stops
clear_margins	mgc	MC	Clear all margins (top, bottom, and sides)
clear_screen	clear	cl	Clear screen and home cursor
clr_bol	el1	cb	Clear to beginning of line, inclusive
clr_eol	el	ce	Clear to end of line
clr_eos	ed	cd	Clear to end of display
code_set_init	csin	ci	Init sequence for multiple codesets
color_names	colornm	Yw	Give name for color #1
column_address	hpa	ch	Horizontal position absolute
command_character	cmdch	CC	Terminal settable cmd character in prototype
create_window	cwin	CW	Define win #1 to go from #2,#3 to #4,#5
cursor_address	cup	cm	Move to row #1 column #2
cursor_down	cud1	do	Down one line
cursor_home	home	ho	Home cursor (if no cup)
cursor_invisible	civis	vi	Make cursor invisible
cursor_left	cub1	le	Move left one space
cursor_mem_address	mrcup	CM	Memory relative cursor addressing
cursor_normal	cnorm	ve	Make cursor appear normal (undo vs/vi)
cursor_right	cuf1	nd	Non-destructive space (cursor or carriage right)
cursor_to_ll	ll	ll	Last line, first column (if no cup)
cursor_up	cuu1	up	Upline (cursor up)
cursor_visible	cvvis	vs	Make cursor very visible
define_bit_image_region	defbi	Yx	Define rectangular bit-image region
define_char	defc	ZE	Define a character in a character set
delete_character	dch1	dc	Delete character
delete_line	dl1	dl	Delete line
device_type	devt	dv	Indicate language/codeset support
dial_phone	dial	DI	Dial phone number #1

display_clock	dclk	DK	Display time-of-day clock
display_pc_char	dispc	S1	Display PC character
dis_status_line	dsl	ds	Disable status line
down_half_line	hd	hd	Half-line down (forward ½ linefeed)
ena_acs	enacs	eA	Enable alternate character set
end_bit_image_region	endbi	Yy	End a bit-image region
enter_alt_charset_mode	smacs	as	Start alternate character set
enter_am_mode	smam	SA	Turn on automatic margins
enter_blink_mode	blink	mb	Turn on blinking
enter_bold_mode	bold	md	Turn on bold (extra bright) mode
enter_ca_mode	smcup	ti	String to begin programs that use cup
enter_delete_mode	smdc	dm	Delete mode (enter)
enter_dim_mode	dim	mh	Turn on half-bright mode
enter_doublewide_mode	swidm	ZF	Enable double wide printing
enter_draft_quality	sdrfq	ZG	Set draft quality print
enter_horizontal_hl_mode	ehhlm		Turn on horizontal highlight mode
enter_insert_mode	smir	im	Insert mode (enter)
enter_italics_mode	sitm	ZH	Enable italics
enter_leftward_mode	slm	ZI	Enable leftward carriage motion
enter_left_hl_mode	elhlm		Turn on left highlight mode
enter_low_hl_mode	elohlm		Turn on low highlight mode
enter_micro_mode	smicm	ZJ	Enable micro motion capabilities
enter_near_letter_quality	snlq	ZK	Set near-letter quality print
enter_normal_quality	snrmq	ZL	Set normal quality print
enter_pc_charset_mode	smpch	S2	Enter PC character display mode
enter_protected_mode	prot	mp	Turn on protected mode
enter_reverse_mode	rev	mr	Turn on reverse video mode
enter_right_hl_mode	erhlm		Turn on right highlight mode
enter_scancode_mode	smsc	S4	Enter PC scancode mode
enter_secure_mode	invis	mk	Turn on blank mode (characters invisible)
enter_shadow_mode	sshm	ZM	Enable shadow printing
enter_standout_mode	smso	so	Begin standout mode
enter_subscript_mode	ssubm	ZN	Enable subscript printing

enter_superscript_mode	ssupm	ZO	Enable superscript printing
enter_top_hl_mode	ethlm		Turn on top highlight mode
enter_underline_mode	smul	us	Start underscore mode
enter_upward_mode	sum	ZP	Enable upward carriage motion
enter_vertical_hl_mode	evhlm		Turn on vertical highlight mode
enter_xon_mode	smxon	SX	Turn on xon/xoff handshaking
erase_chars	ech	ec	Erase #1 characters
exit_alt_charset_mode	rmacs	ae	End alternate character set
exit_am_mode	rmam	RA	Turn off automatic margins
exit_attribute_mode	sgr0	me	Turn off all attributes
exit_ca_mode	rmcup	te	String to end programs that use cup
exit_delete_mode	rmdc	ed	End delete mode
exit_doublewide_mode	rwidm	ZQ	Disable double wide printing
exit_insert_mode	rmir	ei	End insert mode
exit_italics_mode	ritm	ZR	Disable italics
exit_leftward_mode	rlm	ZS	Enable rightward (normal) carriage motion
exit_micro_mode	rmicm	ZT	Disable micro motion capabilities
exit_pc_charset_mode	rmpch	S3	Disable PC character display mode
exit_scancode_mode	rmsc	S5	Disable PC scancode mode
exit_shadow_mode	rshm	ZU	Disable shadow printing
exit_standout_mode	rmso	se	End standout mode
exit_subscript_mode	rsubm	ZV	Disable subscript printing
exit_superscript_mode	rsupm	ZW	Disable superscript printing
exit_underline_mode	rmul	ue	End underscore mode
exit_upward_mode	rum	ZX	Enable downward (normal) carriage motion
exit_xon_mode	rmxon	RX	Turn off xon/xoff handshaking
fixed_pause	pause	PA	Pause for 2-3 seconds
flash_hook	hook	fh	Flash the switch hook
flash_screen	flash	vb	Visible bell (may not move cursor)
form_feed	ff	ff	Hardcopy terminal page eject
from_status_line	fsl	fs	Return from status line
goto_window	wingo	WG	Go to window #1
hangup	hup	HU	Hang-up phone

initialize_color	initc	Ic	Initialize the definition of color
initialize_pair	initp	Ip	Initialize color pair
init_1string	is1	i1	Terminal or printer initialization string
init_2string	is2	is	Terminal or printer initialization string
init_3string	is3	i3	Terminal or printer initialization string
init_file	if	if	Name of initialization file
init_prog	iprog	iP	Path name of program for initialization
insert_character	ich1	ic	Insert character
insert_line	il1	al	Add new blank line
insert_padding	ip	ip	Insert pad after character inserted

The `key_` strings are sent by specific keys. The `key_` descriptions include the macro, defined in `curses.h` [see `curses(5)`], for the code returned by the `curses` routine `getch()` when the key is pressed [see `getch(3X)`].

Variable	CN	TCC	Description
keypad_local	rmkx	ke	Out of "keypad-transmit" mode
keypad_xmit	smkx	ks	Put terminal in "keypad-transmit" mode
key_a1	ka1	K1	KEY_A1, upper left of keypad
key_a3	ka3	K3	KEY_A3, upper right of keypad
key_b2	kb2	K2	KEY_B2, center of keypad
key_backspace	kbs	kb	KEY_BACKSPACE, sent by backspace key
key_beg	kbeg	@1	KEY_BEG, sent by beg(inning) key
key_btab	kcbt	kB	KEY_BTAB, sent by backtab key
key_c1	kc1	K4	KEY_C1, lower left of keypad
key_c3	kc3	K5	KEY_C3, lower right of keypad
key_cancel	kcan	@2	KEY_CANCEL, sent by cancel key
key_catab	ktbc	ka	KEY_CATAB, sent by clear-all-tabs key
key_clear	kclr	kC	KEY_CLEAR, sent by clear-screen or erase key
key_close	kclo	@3	KEY_CLOSE, sent by close key
key_command	kcmd	@4	KEY_COMMAND, sent by cmd (command) key
key_copy	kcpy	@5	KEY_COPY, sent by copy key
key_create	kcrt	@6	KEY_CREATE, sent by create key
key_ctab	kctab	kt	KEY_CTAB, sent by clear-tab key
key_dc	kdch1	kD	KEY_DC, sent by delete-character key

key_dl	kdl1	kL	KEY_DL, sent by delete-line key
key_down	kcud1	kd	KEY_DOWN, sent by terminal down-arrow key
key_eic	krmir	kM	KEY_EIC, sent by rmir or smir in insert mode
key_end	kend	@7	KEY_END, sent by end key
key_enter	kent	@8	KEY_ENTER, sent by enter/send key
key_eol	kel	kE	KEY_EOL, sent by clear-to-end-of-line key
key_eos	ked	kS	KEY_EOS, sent by clear-to-end-of-screen key
key_exit	kext	@9	KEY_EXIT, sent by exit key
key_f0	kf0	k0	KEY_F(0), sent by function key f0
key_f1	kf1	k1	KEY_F(1), sent by function key f1
.	.	.	.
.	.	.	.
.	.	.	.
key_f62	kf62	Fq	KEY_F(62), sent by function key f62
key_f63	kf63	Fr	KEY_F(63), sent by function key f63
key_find	kfnd	@0	KEY_FIND, sent by find key
key_help	khlp	%1	KEY_HELP, sent by help key
key_home	khome	kh	KEY_HOME, sent by home key
key_ic	kich1	kI	KEY_IC, sent by ins-char/enter ins-mode key
key_il	kil1	kA	KEY_IL, sent by insert-line key
key_left	kcub1	kl	KEY_LEFT, sent by terminal left-arrow key
key_ll	kll	kH	KEY_LL, sent by home-down key
key_mark	kmrk	%2	KEY_MARK, sent by mark key
key_message	kmsg	%3	KEY_MESSAGE, sent by message key
key_mouse	kmous	Km	0631, Mouse event has occurred
key_move	kmov	%4	KEY_MOVE, sent by move key
key_next	knxt	%5	KEY_NEXT, sent by next object key
key_npage	knp	kN	KEY_NPAGE, sent by next page key
key_open	kopn	%6	KEY_OPEN, sent by open key
key_options	kopt	%7	KEY_OPTIONS, sent by options key
key_ppage	kpp	kP	KEY_PPAGE, sent by previous page key
key_previous	kprv	%8	KEY_PREVIOUS, sent by previous object key
key_print	kprt	%9	KEY_PRINT, sent by print or copy key

key_redo	krdo	%0	KEY_REDO, sent by redo key
key_reference	kref	&1	KEY_REFERENCE, sent by ref(erence) key
key_refresh	krfr	&2	KEY_REFRESH, sent by refresh key
key_replace	krpl	&3	KEY_REPLACE, sent by replace key
key_restart	krst	&4	KEY_RESTART, sent by restart key
key_resume	kres	&5	KEY_RESUME, sent by resume key
key_right	kcuf1	kr	KEY_RIGHT, sent by terminal right-arrow key
key_save	ksav	&6	KEY_SAVE, sent by save key
key_sbeg	kBEG	&9	KEY_SBEG, sent by shifted beginning key
key_scancel	kCAN	&0	KEY_SCANCEL, sent by shifted cancel key
key_scommand	kCMD	*1	KEY_SCOMMAND, sent by shifted command key
key_scopy	kCPY	*2	KEY_SCOPY, sent by shifted copy key
key_screate	kCRT	*3	KEY_SCREATE, sent by shifted create key
key_sdc	kDC	*4	KEY_SDC, sent by shifted delete-char key
key_sdl	kDL	*5	KEY_SDL, sent by shifted delete-line key
key_select	kslt	*6	KEY_SELECT, sent by select key
key_send	kEND	*7	KEY_SEND, sent by shifted end key
key_seol	kEOL	*8	KEY_SEOL, sent by shifted clear-line key
key_sexit	kEXT	*9	KEY_SEXIT, sent by shifted exit key
key_sf	kind	kF	KEY_SF, sent by scroll-forward/down key
key_sfind	kFND	*0	KEY_SFIND, sent by shifted find key
key_shelp	kHLP	#1	KEY_SHELP, sent by shifted help key
key_shome	kHOM	#2	KEY_SHOME, sent by shifted home key
key_sic	kIC	#3	KEY_SIC, sent by shifted input key
key_sleft	kLFT	#4	KEY_SLEFT, sent by shifted left-arrow key
key_smessage	kMSG	%a	KEY_SMESSAGE, sent by shifted message key
key_smove	kMOV	%b	KEY_SMOVE, sent by shifted move key
key_snext	kNXT	%c	KEY_SNEXT, sent by shifted next key
key_soptions	kOPT	%d	KEY_SOPTIONS, sent by shifted options key
key_sprevious	kPRV	%e	KEY_SPREVIOUS, sent by shifted prev key
key_sprint	kPRT	%f	KEY_SPRINT, sent by shifted print key
key_sr	kri	kR	KEY_SR, sent by scroll-backward/up key
key_sredo	kRDO	%g	KEY_SREDO, sent by shifted redo key

key_sreplace	kRPL	%h	KEY_SREPLACE, sent by shifted replace key
key_sright	kRIT	%i	KEY_SRIGHT, sent by shifted right-arrow key
key_srsume	kRES	%j	KEY_SRSUME, sent by shifted resume key
key_ssave	kSAV	!1	KEY_SSAVE, sent by shifted save key
key_ssuspend	kSPD	!2	KEY_SSUSPEND, sent by shifted suspend key
key_stab	khts	kT	KEY_STAB, sent by set-tab key
key_sundo	kUND	!3	KEY_SUNDO, sent by shifted undo key
key_suspend	kspd	&7	KEY_SUSPEND, sent by suspend key
key_undo	kund	&8	KEY_UNDO, sent by undo key
key_up	kcuu1	ku	KEY_UP, sent by terminal up-arrow key
label_format	fln	Lf	Label format
label_off	rmln	LF	Turn off soft labels
label_on	smln	LO	Turn on soft labels
lab_f0	lf0	l0	Labels on function key f0 if not f0
lab_f1	lf1	l1	Labels on function key f1 if not f1
lab_f10	lf10	la	Labels on function key f10 if not f10
lab_f2	lf2	l2	Labels on function key f2 if not f2
lab_f3	lf3	l3	Labels on function key f3 if not f3
lab_f4	lf4	l4	Labels on function key f4 if not f4
lab_f5	lf5	l5	Labels on function key f5 if not f5
lab_f6	lf6	l6	Labels on function key f6 if not f6
lab_f7	lf7	l7	Labels on function key f7 if not f7
lab_f8	lf8	l8	Labels on function key f8 if not f8
lab_f9	lf9	l9	Labels on function key f9 if not f9
meta_off	rmm	mo	Turn off "meta mode"
meta_on	smm	mm	Turn on "meta mode" (8th bit)
micro_column_address	mhpa	ZY	Like column_address for micro adjustment
micro_down	mcud1	ZZ	Like cursor_down for micro adjustment
micro_left	mcub1	Za	Like cursor_left for micro adjustment
micro_right	mcuf1	Zb	Like cursor_right for micro adjustment
micro_row_address	mvpa	Zc	Like row_address for micro adjustment
micro_up	mcuu1	Zd	Like cursor_up for micro adjustment
mouse_info	minfo	Mi	Mouse status information

newline	nel	nw	Newline (behaves like cr followed by lf)
order_of_pins	porder	Ze	Matches software bits to print-head pins
orig_colors	oc	oc	Set all color(-pair)s to the original ones
orig_pair	op	op	Set default color pair to the original one
pad_char	pad	pc	Pad character (rather than null)
parm_dch	dch	DC	Delete #1 chars
parm_delete_line	dl	DL	Delete #1 lines
parm_down_cursor	cud	DO	Move down #1 lines
parm_down_micro	mcud	Zf	Like parm_down_cursor for micro adjustment
parm_ich	ich	IC	Insert #1 blank chars
parm_index	indn	SF	Scroll forward #1 lines
parm_insert_line	il	AL	Add #1 new blank lines
parm_left_cursor	cub	LE	Move cursor left #1 spaces
parm_left_micro	mcub	Zg	Like parm_left_cursor for micro adjustment
parm_right_cursor	cuf	RI	Move right #1 spaces
parm_right_micro	mcuf	Zh	Like parm_right_cursor for micro adjustment
parm_rindex	rin	SR	Scroll backward #1 lines
parm_up_cursor	cuu	UP	Move cursor up #1 lines
parm_up_micro	mcuu	Zi	Like parm_up_cursor for micro adjustment
pc_term_options	pctrm	S6	PC terminal options
pkey_key	pfkey	pk	Prog funct key #1 to type string #2
pkey_local	pfloc	pl	Prog funct key #1 to execute string #2
pkey_plab	pfxl	xl	Prog key #1 to xmit string #2 and show string #3
pkey_xmit	pfx	px	Prog funct key #1 to xmit string #2
plab_norm	pln	pn	Prog label #1 to show string #2
print_screen	mc0	ps	Print contents of the screen
prtr_non	mc5p	pO	Turn on the printer for #1 bytes
prtr_off	mc4	pf	Turn off the printer
prtr_on	mc5	po	Turn on the printer
pulse	pulse	PU	Select pulse dialing
quick_dial	qdial	QD	Dial phone number #1, without progress detection
remove_clock	rmclk	RC	Remove time-of-day clock
repeat_char	rep	rp	Repeat char #1 #2 times

req_for_input	rfi	RF	Send next input char (for ptys)
req_mouse_pos	reqmp	RQ	Request mouse position report
reset_1string	rs1	r1	Reset terminal completely to sane modes
reset_2string	rs2	r2	Reset terminal completely to sane modes
reset_3string	rs3	r3	Reset terminal completely to sane modes
reset_file	rf	rf	Name of file containing reset string
restore_cursor	rc	rc	Restore cursor to position of last sc
row_address	vpa	cv	Vertical position absolute
save_cursor	sc	sc	Save cursor position
scancode_escape	scesc	S7	Escape for scancode emulation
scroll_forward	ind	sf	Scroll text up
scroll_reverse	ri	sr	Scroll text down
select_char_set	scs	Zj	Select character set
set0_des_seq	s0ds	s0	Shift into codeset 0 (EUC set 0, ASCII)
set1_des_seq	s1ds	s1	Shift into codeset 1
set2_des_seq	s2ds	s2	Shift into codeset 2
set3_des_seq	s3ds	s3	Shift into codeset 3
set_attributes	sgr	sa	Define the video attributes #1-#9
set_a_attributes	sgr1		Define second set of video attributes #1-#6
set_a_background	setab	AB	Set background color using ANSI escape
set_a_foreground	setaf	AF	Set foreground color using ANSI escape
set_background	setb	Sb	Set current background color
set_bottom_margin	smgb	Zk	Set bottom margin at current line
set_bottom_margin_parm	smgbp	Zl	Set bottom margin at line #1 or #2 lines from bottom
set_clock	sclk	SC	Set clock to hours (#1), minutes (#2), seconds (#3)
set_color_band	setcolor	Yz	Change to ribbon color #1
set_color_pair	scp	sp	Set current color pair
set_foreground	setf	Sf	Set current foreground color1
set_left_margin	smgl	ML	Set left margin at current line
set_left_margin_parm	smglp	Zm	Set left (right) margin at column #1 (#2)
set_lr_margin	smglr	ML	Sets both left and right margins
set_page_length	slines	YZ	Set page length to #1 lines

set_pglen_inch	slength	YI	Set page length to #1 hundredth of an inch
set_right_margin	smgr	MR	Set right margin at current column
set_right_margin_parm	smgrp	Zn	Set right margin at column #1
set_tab	hts	st	Set a tab in all rows, current column
set_tb_margin	smgtb	MT	Sets both top and bottom margins
set_top_margin	smgt	Zo	Set top margin at current line
set_top_margin_parm	smgtp	Zp	Set top (bottom) margin at line #1 (#2)
set_window	wind	wi	Current window is lines #1-#2 cols #3-#4
start_bit_image	sbim	Zq	Start printing bit image graphics
start_char_set_def	scsd	Zr	Start definition of a character set
stop_bit_image	rbim	Zs	End printing bit image graphics
stop_char_set_def	rcsd	Zt	End definition of a character set
subscript_characters	subcs	Zu	List of "subscriptable" characters
superscript_characters	supcs	Zv	List of "superscriptable" characters
tab	ht	ta	Tab to next 8-space hardware tab stop
these_cause_cr	docr	Zw	Printing any of these chars causes cr
tone	tone	TO	Select touch tone dialing
to_status_line	tsl	ts	Go to status line, column #1
underline_char	uc	uc	Underscore one char and move past it
up_half_line	hu	hu	Half-line up (reverse ½ linefeed)
user0	u0	u0	User string 0
user1	u1	u1	User string 1
user2	u2	u2	User string 2
user3	u3	u3	User string 3
user4	u4	u4	User string 4
user5	u5	u5	User string 5
user6	u6	u6	User string 6
user7	u7	u7	User string 7
user8	u8	u8	User string 8
user9	u9	u9	User string 9
wait_tone	wait	WA	Wait for dial tone
xoff_character	xoffc	XF	X-off character
xon_character	xonc	XN	X-on character

zero_motion	zerom	Zx	No motion for the subsequent character

Sample Entry

The following entry, which describes the AT&T 610 terminal, is among the more complex entries in the `terminfo` file as of this writing.

```
610|610bct|ATT610|att610|AT&T610;80column;98key keyboard
   am, eslok, hs, mir, msgr, xenl, xon,
   cols#80, it#8, lh#2, lines#24, lw#8, nlab#8, wsl#80,
   acsc=``aaffggjjkkllmmnnooppqqrrssttuuvvwwxxyyzz{{||}}~~,
   bel=^G, blink=\E[5m, bold=\E[1m, cbt=\E[Z,
   civis=\E[?25l, clear=\E[H\E[J, cnorm=\E[?25h\E[?12l,
   cr=\r, csr=\E[%i%p1%d;%p2%dr, cub=\E[%p1%dD, cub1=\b,

 cud=\E[%p1%dB, cud1=\E[B, cuf=\E[%p1%dC, cuf1=\E[C,
   cup=\E[%i%p1%d;%p2%dH, cuu=\E[%p1%dA, cuu1=\E[A,
   cvvis=\E[?12;25h, dch=\E[%p1%dP, dch1=\E[P, dim=\E[2m,
   dl=\E[%p1%dM, dl1=\E[M, ed=\E[J, el=\E[K, el1=\E[1K,
   flash=\E[?5h$<200>\E[?5l, fsl=\E8, home=\E[H, ht=\t,

 ich=\E[%p1%d@, il=\E[%p1%dL, il1=\E[L, ind=\ED, .ind=\ED$<9>,
   invis=\E[8m,
   is1=\E[8;0 | \E[?3;4;5;13;15l\E[13;20l\E[?7h\E[12h\E(B\E)0,
   is2=\E[0m^O, is3=\E(B\E)0, kLFT=\E[\s@, kRIT=\E[\sA,
   kbs=^H, kcbt=\E[Z, kclr=\E[2J, kcub1=\E[D, kcud1=\E[B,
   kcuf1=\E[C, kcuu1=\E[A, kf1=\EOc,
kf10=\ENp,
   kf11=\ENq, kf12=\ENr, kf13=\ENs, kf14=\ENt, kf2=\EOd,
   kf3=\EOe, kf4=\EOf, kf5=\EOg, kf6=\EOh, kf7=\EOi,
   kf8=\EOj, kf9=\ENo, khome=\E[H, kind=\E[S, kri=\E[T,
   ll=\E[24H, mc4=\E[?4i, mc5=\E[?5i, nel=\EE,
   pfxl=\E[%p1%d;%p2%l%02dq%?%p1%{9}%<%t\s\s\sF%p1%1d\s\s\s\s\s
\s\s\s\s\s\s%;%p2%s,

 pln=\E[%p1%d;0;0;0q%p2%:-16.16s, rc=\E8, rev=\E[7m,
   ri=\EM, rmacs=^O, rmir=\E[4l, rmln=\E[2p, rmso=\E[m,
   rmul=\E[m, rs2=\Ec\E[?3l, sc=\E7,
   sgr=\E[0%?%p6%t;1%;%?%p5%t;2%;%?%p2%t;4%;%?%p4%t;5%;
%?%p3%p1% | %t;7%;%?%p7%t;8%;m%?%p9%t^N%e^O%;,
   sgr0=\E[m^O, smacs=^N, smir=\E[4h,
smln=\E[p,
   smso=\E[7m, smul=\E[4m, tsl=\E7\E[25;%i%p1%dx,
```

Types of Capabilities in the Sample Entry

The sample entry shows the formats for the three types of `terminfo` capabilities listed: Boolean, numeric, and string. All capabilities specified in the `terminfo` source file must be followed by commas, including the last capability in the source file. In `terminfo` source files, capabilities are referenced by their capability names (as shown in the previous tables).

Boolean Capabilities

Boolean capabilities are specified simply by their comma separated cap names.

A boolean capability is true if its capname is present in the entry, and false if its capname is not present in the entry.

The @ character following a capname is used to explicitly declare that a boolean capability is false, in situations described in section 1-16: "Similar Terminals" (see below).

Numeric Capabilities

Numeric capabilities are followed by the character # and then a positive integer value. Thus, in the sample, `cols` (which shows the number of columns available on a device) is assigned the value `80` for the AT&T 610. (Values for numeric capabilities may be specified in decimal, octal, or hexadecimal, using normal C programming language conventions.)

String Capabilities

Finally, string-valued capabilities such as `el` (clear to end of line sequence) are listed by a two- to five-character capname, an =, and a string ended by the next occurrence of a comma.

A delay in milliseconds may appear anywhere in such a capability, preceded by `$` and enclosed in angle brackets, as in `el=\EK$<3>`. The `curses` implementation achieves delays by outputting to the terminal an appropriate number of system-defined padding characters. The `tputs()` function provides delays when used to send such a capability to the terminal.

The delay can be any of the following: a number, a number followed by an asterisk, such as `5*`, a number followed by a slash, such as `5/`, or a number followed by both, such as `5*/`.

- A`*` shows that the padding required is proportional to the number of lines affected by the operation, and the amount given is the per-affected-unit padding required. (In the case of insert characters, the factor is still the number of lines affected. This is always 1 unless the device has `in` and the software uses it.) When a `*` is specified, it is sometimes useful to give a delay of the form `3.5` to specify a delay per unit to tenths of milliseconds. (Only one decimal place is allowed.)
- A`/` indicates that the delay is mandatory and padding characters are transmitted regardless of the setting of `xon`. If `/` is not specified or if a device has `xon` defined, the delay information is advisory and is only used for cost estimates or when the device is in raw mode. However, any delay specified for `bel` or `flash` is treated as mandatory.

The following notation is valid in `terminfo` source files for specifying special characters:

Notation	Represents Character
^*x*	Control-*x* (for any appropriate *x*)
\a	Alert
\b	Backspace
\E or \e	An ESCAPE character
\f	Form feed
\l	Linefeed
\n	Newline
\r	Carriage return
\s	Space
\t	Tab
\^	Caret (^)
\\	Backslash (\)
\,	Comma (,)
\:	Colon (:)
\0	Null *
nnn	Any character, specified as three octal digits

* `\0` will actually produce `\200`, which does not terminate a string but behaves as a null character on most devices, providing CS7 is specified [see `stty`(1)].

Commented-out Capabilities

Sometimes individual capabilities must be commented out. To do this, put a period before the capability name. For example, see the second `ind` in the example above. Note that capabilities are defined in a left-to-right order and, therefore, a prior definition will override a later definition.

Preparing Descriptions

The most effective way to prepare a device description is by imitating the description of a similar device in `terminfo` and building up a description gradually, using partial descriptions with `vi` to check that they are correct. Be aware that a very unusual device may expose deficiencies in the ability of the `terminfo` file to describe it or the inability of `vi` to work with that device. To test a new device description, set the environment variable `TERMINFO` to the pathname of a directory containing the compiled description you are working on and programs will look there rather than in `/usr/share/lib/terminfo`. To get the padding for insert-line correct (if the device manufacturer did not document it) a severe test is to comment out `xon`, edit a large file at 9600 baud with `vi`, delete 16 or so lines from the middle of the screen, and then press the `u` key several times quickly. If the display is corrupted, more padding is usually needed. A similar test can be used for insert character.

Section 1-1: Basic Capabilities

The number of columns on each line for the device is given by the `cols` numeric capability. If the device has a screen, then the number of lines on the screen is given by the `lines` capability. If the device wraps around to the beginning of the next line when it reaches the right margin, then it should have the `am` capability. If the terminal can clear its screen, leaving the cursor in the home position, then this is given by the `clear` string capability. If the terminal overstrikes (rather than clearing a position when a character is struck over) then it should have the `os` capability. If the device is a printing terminal, with no soft copy unit, specify both `hc` and `os`. If there is a way to move the cursor to the left edge of the current row, specify this as `cr`. (Normally this will be carriage return, <CTRL-M>.) If there is a way to produce an audible signal (such as a bell or a beep), specify it as `bel`. If, like most devices, the device uses the xon-xoff flow-control protocol, specify `xon`.

If there is a way to move the cursor one position to the left (such as backspace), that capability should be given as `cub1`. Similarly, sequences to move to the right, up, and down should be given as `cuf1`, `cuu1`, and `cud1`, respectively. These local cursor motions must not alter the text they pass over; for example, you would not normally use `cuf1=\s` because the space would erase the character moved over.

A very important point here is that the local cursor motions encoded in `terminfo` are undefined at the left and top edges of a screen terminal. Programs should never attempt to backspace around the left edge, unless `bw` is specified, and should never attempt to go up locally off the top. To scroll text up, a program goes to the bottom left corner of the screen and sends the `ind` (index) string.

To scroll text down, a program goes to the top left corner of the screen and sends the `ri` (reverse index) string. The strings `ind` and `ri` are undefined when not on their respective corners of the screen.

Parameterized versions of the scrolling sequences are `indn` and `rin`. These versions have the same semantics as `ind` and `ri`, except that they take one parameter and scroll the number of lines specified by that parameter. They are also undefined except at the appropriate edge of the screen.

The `am` capability tells whether the cursor sticks at the right edge of the screen when text is output, but this does not necessarily apply to a `cuf1` from the last column. Backward motion from the left edge of the screen is possible only when `bw` is specified. In this case, `cub1` will move to the right edge of the previous row. If `bw` is not given, the effect is undefined. This is useful for drawing a box around the edge of the screen, for example. If the device has switch selectable automatic margins, `am` should be specified in the `terminfo` source file. In this case, initialization strings should turn on this option, if possible. If the device has a command that moves to the first column of the next line, that command can be given as `nel` (newline). It does not matter if the command clears the remainder of the current line, so if the device has no `cr` and `lf` it may still be possible to craft a working `nel` out of one or both of them.

These capabilities suffice to describe hardcopy and screen terminals. Thus the AT&T 5320 hardcopy terminal

is described as follows:

```
5320|att5320|AT&T 5320 hardcopy terminal,
   am, hc, os,
   cols#132,
   bel=^G, cr=\r, cub1=\b, cnd1=\n,
   dch1=\E[P, dl1=\E[M,
   ind=\n,
```

while the Lear Siegler ADM-3 is described as

```
adm3|lsi adm3,
   am, bel=^G, clear=^Z, cols#80, cr=^M, cub1=^H,
   cud1=^J, ind=^J, lines#24,
```

Section 1-2: Parameterized Strings

Cursor addressing and other strings requiring parameters are described by a parameterized string capability, with `printf`-like escapes (`%`*x*) in it. For example, to address the cursor, the `cup` capability is given, using two parameters: the row and column to address to. (Rows and columns are numbered from zero and refer to the physical screen visible to the user, not to any unseen memory.) If the terminal has memory relative cursor addressing, that can be indicated by `mrcup`.

The parameter mechanism uses a stack and special `%` codes to manipulate the stack in the manner of Reverse Polish Notation (postfix). Typically a sequence will push one of the parameters onto the stack and then print it in some format. Often more complex operations are necessary. Operations are in postfix form with the operands in the usual order. That is, to subtract 5 from the first parameter, one would use `%p1%{5}%-`.

The `%` encodings have the following meanings:

`%%` Outputs `%`

`%`[`[:`]*flags*][*width*[*.precision*]][`doxXs`]
as in `printf`, flags are [`-+#`] and space

`%c` Print pop gives `%c`

`%p`[`1-9`]
Push *i*th argument

`%P`[`a-z`]
Set dynamic variable [`a-z`] to pop

`%g`[`a-z`]
Get dynamic variable [`a-z`] and push it

`%P`[`A-Z`]
Set static variable [`a-z`] to pop

`%g`[`A-Z`]
Get static variable [`a-z`] and push it

`%'`*c*`'` Push `char` constant *c*

`%{`*nn*`}` Push decimal constant *nn*

`%l` pop a string address and push its length

`%+ %- %* %/ %m`
Arithmetic (`%m` is modulus): push(pop $integer_2$ op pop $integer_1$)

`%& %| %^`
Bit operations: push(pop $integer_2$ op pop $integer_1$)

`%= %> %<`
Logical operations: push(pop $integer_2$ op pop $integer_1$)

`%A %O` Logical operations: and, or

`%! %~` Unary operations: push(op pop)

`%i` (For ANSI terminals) Add 1 to first parameter, if one parameter present, or first two parameters, if more than one parameter present

`%?` *expr* `%t` *thenpart* `%e` *elsepart* `%;`
if-then-else, `%e` *elsepart* is optional; else-if's are possible ala Algol 68: `%? c1 %t b1 %e c2 %t b2 %e c3 %t b3 %e c4 %t b4 %e b5%.`

c_i are conditions, b_i are bodies.

If the – flag is used with `%[doxXs]`, then a colon (`:`) must be placed between the `%` and the – to differentiate the flag from the binary `%–` operator, for example `%:–16.16s`.

Consider the Hewlett-Packard 2645, which, to get to row 3 and column 12, needs to be sent `\E&a12c03Y` padded for 6 milliseconds. Note that the order of the rows and columns is inverted here, and that the row and column are zero-padded as two digits. Thus its `cup` capability is:

```
cup=\E&a%p2%2.2dc%p1%2.2dY$<6>
```

The Micro-Term ACT-IV needs the current row and column sent preceded by a `^T`, with the row and column simply encoded in binary, `cup=^T%p1%c%p2%c`. Devices that use `%c` need to be able to backspace the cursor (`cub1`), and to move the cursor up one line on the screen (`cuu1`). This is necessary because it is not always safe to transmit `\n`, `^D`, and `\r`, as the system may change or discard them. (The library routines dealing with `terminfo` set tty modes so that tabs are never expanded, so `\t` is safe to send. This turns out to be essential for the Ann Arbor 4080.)

A final example is the LSI ADM-3a, which uses row and column offset by a blank character, thus `cup=\E=%p1%'\s'%+%c%p2%'\s'%+%c`. After sending `\E=`, this pushes the first parameter, pushes the ASCII value for a space (32), adds them (pushing the sum on the stack in place of the two previous values), and outputs that value as a character. Then the same is done for the second parameter. More complex arithmetic is possible using the stack.

Section 1-3: Cursor Motions

If the terminal has a fast way to home the cursor (to very upper left corner of screen) then this can be given as `home`; similarly a fast way of getting to the lower left-hand corner can be given as `ll`; this may involve going up with `cuu1` from the home position, but a program should never do this itself (unless `ll` does) because it can make no assumption about the effect of moving up from the home position. Note that the home position is the same as addressing to (0,0): to the top left corner of the screen, not of memory. (Thus, the `\EH` sequence on Hewlett-Packard terminals cannot be used for `home` without losing some of the other features on the terminal.)

If the device has row or column absolute-cursor addressing, these can be given as single parameter capabilities `hpa` (horizontal position absolute) and `vpa` (vertical position absolute). Sometimes these are shorter than the more general two-parameter sequence (as with the Hewlett-Packard 2645) and can be used in preference to `cup`. If there are parameterized local motions (for example, move *n* spaces to the right) these can be given as `cud`, `cub`, `cuf`, and `cuu` with a single parameter indicating how many spaces to move. These are primarily useful if the device does not have `cup`, such as the Tektronix 4025.

If the device needs to be in a special mode when running a program that uses these capabilities, the codes to enter and exit this mode can be given as `smcup` and `rmcup`. This arises, for example, from terminals, such as the Concept, with more than one page of memory. If the device has only memory relative cursor addressing and not screen relative cursor addressing, a one screen-sized window must be fixed into the device for cursor addressing to work properly. This is also used for the Tektronix 4025, where `smcup` sets the command character to be the one used by `terminfo`. If the `smcup` sequence will not restore the screen after an `rmcup` sequence is output (to the state prior to outputting `rmcup`), specify `nrrmc`.

Section 1-4: Area Clears

If the terminal can clear from the current position to the end of the line, leaving the cursor where it is, this

should be given as el. If the terminal can clear from the beginning of the line to the current position inclusive, leaving the cursor where it is, this should be given as el1. If the terminal can clear from the current position to the end of the display, then this should be given as ed. ed is only defined from the first column of a line. (Thus, it can be simulated by a request to delete a large number of lines, if a true ed is not available.)

Section 1-5: Insert/Delete Line

If the terminal can open a new blank line before the cursor line, this should be given as il1; this is done only from the first position of a line. The cursor must then appear on the new blank line. If the terminal can delete the line on which the cursor is positioned, this should be given as dl1; this is only done from the first position on the line to be deleted. Versions of il1 and dl1 which take a single parameter and insert or delete that many lines can be given as il and dl.

If the terminal has a settable destructive scrolling region (like the VT100) the command to set this can be described with the csr capability, which takes two parameters: the top and bottom lines of the scrolling region. The cursor position is, alas, undefined after using this command. It is possible to get the effect of insert or delete line using this command – the sc and rc (save and restore cursor) commands are also useful. Inserting lines at the top or bottom of the screen can also be done using ri or ind on many terminals without a true insert/delete line, and is often faster even on terminals with those features.

To determine whether a terminal has destructive scrolling regions or non-destructive scrolling regions, create a scrolling region in the middle of the screen, place data on the bottom line of the scrolling region, move the cursor to the top line of the scrolling region, and do a reverse index (ri) followed by a delete line (dl1) or index (ind). If the data that was originally on the bottom line of the scrolling region was restored into the scrolling region by the dl1 or ind, then the terminal has non-destructive scrolling regions. Otherwise, it has destructive scrolling regions. Do not specify csr if the terminal has non-destructive scrolling regions, unless ind, ri, indn, rin, dl, and dl1 all simulate destructive scrolling.

If the terminal has the ability to define a window as part of memory, which all commands affect, it should be given as the parameterized string wind. The four parameters are the starting and ending lines in memory and the starting and ending columns in memory, in that order.

If the terminal can retain display memory above, then the da capability should be given; if display memory can be retained below, then db should be given. These indicate that deleting a line or scrolling a full screen may bring non-blank lines up from below or that scrolling back with ri may bring down non-blank lines.

Section 1-6: Insert/Delete Character

There are two basic kinds of intelligent terminals with respect to insert/delete character operations which can be described using terminfo. The most common insert/delete character operations affect only the characters on the current line and shift characters off the end of the line rigidly. Other terminals, such as the Concept 100 and the Perkin Elmer Owl, make a distinction between typed and untyped blanks on the screen, shifting upon an insert or delete only to an untyped blank on the screen which is either eliminated, or expanded to two untyped blanks. You can determine the kind of terminal you have by clearing the screen and then typing text separated by cursor motions. Type abc□□□□def using local cursor motions (not spaces) between the abc and the def. Then position the cursor before the abc and put the terminal in insert mode. If typing characters causes the rest of the line to shift rigidly and characters to fall off the end, then your terminal does not distinguish between blanks and untyped positions. If the abc shifts over to the def which then move together around the end of the current line and onto the next as you insert, you have the second type of terminal and should give the capability in, which stands for insert null. While these are two logically separate attributes (one line versus multiline insert mode and special treatment of untyped spaces) we have seen no terminals whose insert mode cannot be described with the single attribute.

terminfo can describe both terminals that have an insert mode and terminals which send a simple sequence to open a blank position on the current line. Give as smir the sequence to get into insert mode. Give as rmir the sequence to leave insert mode. Now give as ich1 any sequence needed to be sent just before sending the character to be inserted. Most terminals with a true insert mode will not give ich1; terminals that send a sequence to open a screen position should give it here. (If your terminal has both, insert mode is usually preferable to ich1. Do not give both unless the terminal actually requires both to be used in

combination.) If post-insert padding is needed, give this as a number of milliseconds padding in `ip` (a string option). Any other sequence which may need to be sent after an insert of a single character may also be given in `ip`. If your terminal needs both to be placed into an insert mode and a special code to precede each inserted character, then both `smir`/`rmir` and `ich1` can be given, and both will be used. The `ich` capability, with one parameter, *n*, will insert *n* blanks.

If padding is necessary between characters typed while not in insert mode, give this as a number of milliseconds padding in `rmp`.

It is occasionally necessary to move around while in insert mode to delete characters on the same line (for example, if there is a tab after the insertion position). If your terminal allows motion while in insert mode you can give the capability `mir` to speed up inserting in this case. Omitting `mir` will affect only speed. Some terminals (notably Datamedia's) must not have `mir` because of the way their insert mode works.

Finally, you can specify `dch1` to delete a single character, `dch` with one parameter, *n*, to delete *n* characters, and delete mode by giving `smdc` and `rmdc` to enter and exit delete mode (any mode the terminal needs to be placed in for `dch1` to work).

A command to erase *n* characters (equivalent to outputting *n* blanks without moving the cursor) can be given as `ech` with one parameter.

Section 1-7: Highlighting, Underlining, and Visible Bells

Your device may have one or more kinds of display attributes that allow you to highlight selected characters when they appear on the screen. The following display modes (shown with the names by which they are set) may be available: a blinking screen (`blink`), bold or extra-bright characters (`bold`), dim or half-bright characters (`dim`), blanking or invisible text (`invis`), protected text (`prot`), a reverse-video screen (`rev`), and an alternate character set (`smacs` to enter this mode and `rmacs` to exit it). (If a command is necessary before you can enter alternate character set mode, give the sequence in `enacs` or enable alternate-character-set mode.) Turning on any of these modes singly may or may not turn off other modes.

`sgr0` should be used to turn off all video enhancement capabilities. It should always be specified because it represents the only way to turn off some capabilities, such as `dim` or `blink`.

You should choose one display method as *standout mode* and use it to highlight error messages and other kinds of text to which you want to draw attention. Choose a form of display that provides strong contrast but that is easy on the eyes. (We recommend reverse-video plus half-bright or reverse-video alone.) The sequences to enter and exit standout mode are given as `smso` and `rmso`, respectively. If the code to change into or out of standout mode leaves one or even two blank spaces on the screen, as the TVI 912 and Teleray 1061 do, then `xmc` should be given to tell how many spaces are left.

Sequences to begin underlining and end underlining can be specified as `smul` and `rmul`, respectively. If the device has a sequence to underline the current character and to move the cursor one space to the right (such as the Micro-Term MIME), this sequence can be specified as `uc`.

Terminals with the "magic cookie" glitch (`xmc`) deposit special "cookies" when they receive mode-setting sequences, which affect the display algorithm rather than having extra bits for each character. Some terminals, such as the Hewlett-Packard 2621, automatically leave standout mode when they move to a new line or the cursor is addressed. Programs using standout mode should exit standout mode before moving the cursor or sending a newline, unless the `msgr` capability, asserting that it is safe to move in standout mode, is present.

If the terminal has a way of flashing the screen to indicate an error quietly (a bell replacement), then this can be given as `flash`; it must not move the cursor. A good flash can be done by changing the screen into reverse video, pad for 200 ms, then return the screen to normal video.

If the cursor needs to be made more visible than normal when it is not on the bottom line (to make, for example, a non-blinking underline into an easier to find block or blinking underline) give this sequence as `cvvis`. The boolean `chts` should also be given. If there is a way to make the cursor completely invisible, give that as `civis`. The capability `cnorm` should be given which undoes the effects of either of these modes.

If your terminal generates underlined characters by using the underline character (with no special sequences needed) even though it does not otherwise overstrike characters, then you should specify the capability `ul`.

For devices on which a character overstriking another leaves both characters on the screen, specify the capability `os`. If overstrikes are erasable with a blank, then this should be indicated by specifying `eo`.

If there is a sequence to set arbitrary combinations of modes, this should be given as `sgr` (set attributes), taking nine parameters. Each parameter is either `0` or non-zero, as the corresponding attribute is on or off. The nine parameters are shown below. Not all modes need to be supported by `sgr`; only those for which corresponding separate attribute commands exist should be supported. For example, let's assume that the terminal in question needs the following escape sequences to turn on various modes.

tparm Argument	Attribute	Escape Sequence
	none	`\E[0m`
p1	standout	`\E[0;4;7m`
p2	underline	`\E[0;3m`
p3	reverse	`\E[0;4m`
p4	blink	`\E[0;5m`
p5	dim	`\E[0;7m`
p6	bold	`\E[0;3;4m`
p7	invis	`\E[0;8m`
p8	protect	not available
p9	altcharset	`^O` (off) ^N (on)

Each escape sequence requires a `0` to turn off other modes before turning on its own mode. Also note that, as suggested above, *standout* is set up to be the combination of *reverse* and *dim*. Also, because this terminal has no *bold* mode, *bold* is set up as the combination of *reverse* and *underline*. In addition, to allow combinations, such as *underline*+*blink*, the sequence to use would be `\E[0;3;5m`. The terminal doesn't have *protect* mode, either, but that cannot be simulated in any way, so `p8` is ignored. The *altcharset* mode is different in that it is either `^O` or `^N`, depending on whether it is off or on. If all modes were to be turned on, the sequence would be `\E[0;3;4;5;7;8m^N`.

Now look at when different sequences are output. For example, `;3` is output when either `p2` or `p6` is true, that is, if either *underline* or *bold* modes are turned on. Writing out the above sequences, along with their dependencies, gives the following:

Sequence	**When To Output**	**terminfo** Translation
`\E[0`	always	`\E[0`
`;3`	if `p2` or `p6`	`%?%p2%p6%|%t;3%;`
`;4`	if `p1` or `p3` or `p6`	`%?%p1%p3%|%p6%|%t;4%;`
`;5`	if `p4`	`%?%p4%t;5%;`
`;7`	if `p1` or `p5`	`%?%p1%p5%|%t;7%;`
`;8`	if `p7`	`%?%p7%t;8%;`
`m`	always	`m`
`^N` or ^O	if `p9` `^N`, else `^O`	`%?%p9%t^N%e^O%;`

Putting this all together into the `sgr` sequence gives:

```
sgr=\E[0%?%p2%p6%|%t;3%;%?%p1%p3%|%p6%□
       |%t;4%;%?%p5%t;5%;%?%p1%p5%□
       |%t;7%;%?%p7%t;8%;m%?%p9%t^N%e^O%;,
```

Remember that sgr and sgr0 must always be specified.

Section 1-8: Keypad

If the device has a keypad that transmits sequences when the keys are pressed, this information can also be specified. It is not possible to handle devices where the keypad only works in local (this applies, for example, to the unshifted Hewlett-Packard 2621 keys). If the keypad can be set to transmit or not transmit, specify these sequences as smkx and rmkx. Otherwise the keypad is assumed to always transmit.

The sequences sent by the left arrow, right arrow, up arrow, down arrow, and home keys can be given as kcub1, kcuf1, kcuu1, kcud1, and khome, respectively. If there are function keys such as f0, f1, ..., f63, the sequences they send can be specified as kf0, kf1, ..., kf63. If the first 11 keys have labels other than the default f0 through f10, the labels can be given as lf0, lf1, ..., lf10. The codes transmitted by certain other special keys can be given: kll (home down), kbs (backspace), ktbc (clear all tabs), kctab (clear the tab stop in this column), kclr (clear screen or erase key), kdch1 (delete character), kdl1 (delete line), krmir (exit insert mode), kel (clear to end of line), ked (clear to end of screen), kich1 (insert character or enter insert mode), kil1 (insert line), knp (next page), kpp (previous page), kind (scroll forward/down), kri (scroll backward/up), khts (set a tab stop in this column). In addition, if the keypad has a 3 by 3 array of keys including the four arrow keys, the other five keys can be given as ka1, ka3, kb2, kc1, and kc3. These keys are useful when the effects of a 3 by 3 directional pad are needed. Further keys are defined above in the capabilities list.

Strings to program function keys can be specified as pfkey, pfloc, and pfx. A string to program screen labels should be specified as pln. Each of these strings takes two parameters: a function key identifier and a string to program it with. pfkey causes pressing the given key to be the same as the user typing the given string; pfloc causes the string to be executed by the terminal in local mode; and pfx causes the string to be transmitted to the computer. The capabilities nlab, lw and lh define the number of programmable screen labels and their width and height. If there are commands to turn the labels on and off, give them in smln and rmln. smln is normally output after one or more pln sequences to make sure that the change becomes visible.

Section 1-9: Tabs and Initialization

If the device has hardware tabs, the command to advance to the next tab stop can be given as ht (usually <CTRL-I>). A backtab command that moves leftward to the next tab stop can be given as cbt. By convention, if tty modes show that tabs are being expanded by the computer rather than being sent to the device, programs should not use ht or cbt (even if they are present) because the user may not have the tab stops properly set. If the device has hardware tabs that are initially set every *n* spaces when the device is powered up, the numeric parameter it is given, showing the number of spaces the tabs are set to. This is normally used by tput init [see tput(1)] to determine whether to set the mode for hardware tab expansion and whether to set the tab stops. If the device has tab stops that can be saved in nonvolatile memory, the terminfo description can assume that they are properly set. If there are commands to set and clear tab stops, they can be given as tbc (clear all tab stops) and hts (set a tab stop in the current column of every row).

Other capabilities include: is1, is2, and is3, initialization strings for the device; iprog, the path name of a program to be run to initialize the device; and if, the name of a file containing long initialization strings. These strings are expected to set the device into modes consistent with the rest of the terminfo description. They must be sent to the device each time the user logs in and be output in the following order: run the program iprog; output is1; output is2; set the margins using mgc, smgl and smgr; set the tabs using tbc and hts; print the file if; and finally output is3. This is usually done using the init option of tput.

Most initialization is done with is2. Special device modes can be set up without duplicating strings by putting the common sequences in is2 and special cases in is1 and is3. Sequences that do a reset from a totally unknown state can be given as rs1, rs2, rf, and rs3, analogous to is1, is2, is3, and if. (The method using files, if and rf, is used for a few terminals, from /usr/share/lib/tabset/*; however, the recommended method is to use the initialization and reset strings.) These strings are output by tput reset, which is used when the terminal gets into a wedged state. Commands are normally placed in rs1, rs2, rs3,

and rf only if they produce annoying effects on the screen and are not necessary when logging in. For example, the command to set a terminal into 80-column mode would normally be part of is2, but on some terminals it causes an annoying glitch on the screen and is not normally needed because the terminal is usually already in 80-column mode.

If a more complex sequence is needed to set the tabs than can be described by using tbc and hts, the sequence can be placed in is2 or if.

Any margin can be cleared with mgc. (For instructions on how to specify commands to set and clear margins, refer to the section **PRINTER CAPABILITIES**.)

Section 1-10: Delays

Certain capabilities control padding in the tty driver. These are primarily needed by hardcopy terminals, and are used by tput init to set tty modes appropriately. Delays embedded in the capabilities cr, ind, cub1, ff, and tab can be used to set the appropriate delay bits to be set in the tty driver. If pb (padding baud rate) is given, these values can be ignored at baud rates below the value of pb.

Section 1-11: Status Lines

If the terminal has an extra status line that is not normally used by software, this fact can be indicated. If the status line is viewed as an extra line below the bottom line, into which one can cursor address normally (such as the Heathkit h19's 25th line, or the 24th line of a VT100 which is set to a 23-line scrolling region), the capability hs should be given. Special strings that go to a given column of the status line and return from the status line can be given as tsl and fsl. (fsl must leave the cursor position in the same place it was before tsl. If necessary, the sc and rc strings can be included in tsl and fsl to get this effect.) The capability tsl takes one parameter, which is the column number of the status line the cursor is to be moved to.

If escape sequences and other special commands, such as tab, work while in the status line, the flag eslok can be given. A string which turns off the status line (or otherwise erases its contents) should be given as dsl. If the terminal has commands to save and restore the position of the cursor, give them as sc and rc. The status line is normally assumed to be the same width as the rest of the screen (for example, cols). If the status line is a different width (possibly because the terminal does not allow an entire line to be loaded) the width, in columns, can be indicated with the numeric parameter wsl.

Section 1-12: Line Graphics

If the device has a line drawing alternate character set, the mapping of glyph to character would be given in acsc. The definition of this string is based on the alternate character set used in the DEC VT100 terminal, extended slightly with some characters from the AT&T 4410v1 terminal.

Glyph Name	vt100+ Character
Arrow pointing right	+
Arrow pointing left	,
Arrow pointing down	.
Solid square block	0
Lantern symbol	I
Arrow pointing up	-
Diamond	`
Checker board (stipple)	a
Degree symbol	f
Plus/minus	g
Board of squares	h

Lower right corner	j
Upper right corner	k
Upper left corner	l
Lower left corner	m
Plus	n
Scan line 1	o
Horizontal line	q
Scan line 9	s
□	
Left tee (\z|–)	t
Right tee (–|)	u
Bottom tee ()	v
Top tee ()	w
Vertical line	x
Bullet	~

The best way to describe a new device's line graphics set is to add a third column to the above table with the characters for the new device that produce the appropriate glyph when the device is in the alternate character set mode. For example,

Glyph Name	vt100+ Character	Character Used on New Device
Upper left corner	l	R
Lower left corner	m	F
Upper right corner	k	T
Lower right corner	j	G
Horizontal line	q	,
Vertical line	x	.

Now write down the characters left to right, as in `acsc=lRmFkTjGq\,x.`.

In addition, `terminfo` allows you to define multiple character sets. Refer to section 2-5 for details.

Section 1-13: Color Manipulation

Let us define two methods of color manipulation: the Tektronix method and the HP method. The Tektronix method uses a set of N predefined colors (usually 8) from which a user can select current foreground and background colors. Thus a terminal can support up to N colors mixed into $N*N$ color pairs to be displayed on the screen at the same time. When using an HP method the user cannot define the foreground independently of the background, or vice-versa. Instead, the user must define an entire color pair at once. Up to M color pairs, made from $2*M$ different colors, can be defined this way. Most existing color terminals belong to one of these two classes of terminals.

The numeric variables `colors` and `pairs` define the number of colors and color pairs that can be displayed on the screen at the same time. If a terminal can change the definition of a color (for example, the Tektronix 4100 and 4200 series terminals), this should be specified with `ccc` (can change color). To change the definition of a color (Tektronix 4200 method), use `initc` (initialize color). It requires four arguments: color

number (ranging from 0 to `colors`–1) and three RGB (red, green, and blue) values or three HLS colors (Hue, Lightness, Saturation). Ranges of RGB and HLS values are terminal dependent.

Tektronix 4100 series terminals only use HLS color notation. For such terminals (or dual-mode terminals to be operated in HLS mode) one must define a boolean variable `hls`; that would instruct the `curses init_color` routine to convert its RGB arguments to HLS before sending them to the terminal. The last three arguments to the `initc` string would then be HLS values.

If a terminal can change the definitions of colors, but uses a color notation different from RGB and HLS, a mapping to either RGB or HLS must be developed.

If the terminal supports ANSI escape sequences to set background and foreground, they should be coded as `setab` and `setaf`, respectively. If the terminal supports other escape sequences to set background and foreground, they should be coded as `setb` and `setf`, respectively. The `vidputs()` function and the refresh functions use `setab` and `setaf` if they are defined. Each of these capabilities requires one argument: the number of the color. By convention, the first eight colors (0–7) map to, in order: black, red, green, yellow, blue, magenta, cyan, white. However, color re-mapping may occur or the underlying hardware may not support these colors. Mappings for any additional colors supported by the device (that is, to numbers greater than 7) are at the discretion of the `terminfo` entry writer.

To initialize a color-pair (HP method), use `initp` (initialize pair). It requires seven arguments: the number of a color-pair (range=0 to `pairs`–1), and six RGB values: three for the foreground followed by three for the background. (Each of these groups of three should be in the order RGB.) When `initc` or `initp` are used, RGB or HLS arguments should be in the order "red, green, blue" or "hue, lightness, saturation"), respectively. To make a color-pair current, use `scp` (set color-pair). It takes one argument, the number of a color-pair.

Some terminals (for example, most color terminal emulators for PCs) erase areas of the screen with current background color. In such cases, `bce` (background color erase) should be defined. The variable `op` (original pair) contains a sequence for setting the foreground and the background colors to what they were at the terminal start-up time. Similarly, `oc` (original colors) contains a control sequence for setting all colors (for the Tektronix method) or color pairs (for the HP method) to the values they had at the terminal start-up time.

Some color terminals substitute color for video attributes. Such video attributes should not be combined with colors. Information about these video attributes should be packed into the `ncv` (no color video) variable. There is a one-to-one correspondence between the nine least significant bits of that variable and the video attributes. The following table depicts this correspondence.

Attribute	Bit Position	Decimal Value	Characteristic That Sets
`WA_STANDOUT`	0	1	`sgr`, parameter 1
`WA_UNDERLINE`	1	2	`sgr`, parameter 2
`WA_REVERSE`	2	4	`sgr`, parameter 3
`WA_BLINK`	3	8	`sgr`, parameter 4
`WA_DIM`	4	16	`sgr`, parameter 5
`WA_BOLD`	5	32	`sgr`, parameter 6
`WA_INVIS`	6	64	`sgr`, parameter 7
`WA_PROTECT`	7	128	`sgr`, parameter 8
`WA_ALTCHARSET`	8	256	`sgr`, parameter 9
`WA_HORIZONTAL`	9	512	`sgr1`, parameter 1
`WA_LEFT`	10	1024	`sgr1`, parameter 2
`WA_LOW`	11	2048	`sgr1`, parameter 3
`WA_RIGHT`	12	4096	`sgr1`, parameter 4

WA_TOP	13	8192	sgr1, parameter 5
WA_VERTICAL	14	16384	sgr1, parameter 6

When a particular video attribute should not be used with colors, the corresponding ncv bit should be set to 1; otherwise it should be set to zero. To determine the information to pack into the ncv variable, you must add together the decimal values corresponding to those attributes that cannot coexist with colors. For example, if the terminal uses colors to simulate reverse video (bit number 2 and decimal value 4) and bold (bit number 5 and decimal value 32), the resulting value for ncv will be 36 (4 + 32).

Section 1-14: Miscellaneous

If the terminal requires other than a null (zero) character as a pad, then this can be given as pad. Only the first character of the pad string is used. If the terminal does not have a pad character, specify npc.

If the terminal can move up or down half a line, this can be indicated with hu (half-line up) and hd (half-line down). This is primarily useful for superscripts and subscripts on hardcopy terminals. If a hardcopy terminal can eject to the next page (form feed), give this as ff (usually <CTRL-L>).

If there is a command to repeat a given character a given number of times (to save time transmitting a large number of identical characters) this can be indicated with the parameterized string rep. The first parameter is the character to be repeated and the second is the number of times to repeat it. Thus, tparm(repeat_char, 'x', 10) is the same as xxxxxxxxxx.

If the terminal has a settable command character, such as the Tektronix 4025, this can be indicated with cmdch. A prototype command character is chosen which is used in all capabilities. This character is given in the cmdch capability to identify it. The following convention is supported on some UNIX systems: If the environment variable CC exists, all occurrences of the prototype character are replaced with the character in CC.

Terminal descriptions that do not represent a specific kind of known terminal, such as *switch*, *dialup*, *patch*, and *network*, should include the gn (generic) capability so that programs can complain that they do not know how to talk to the terminal. (This capability does not apply to *virtual* terminal descriptions for which the escape sequences are known.) If the terminal is one of those supported by the Reliant UNIX system virtual terminal protocol, the terminal number can be given as vt. A line-turn-around sequence to be transmitted before doing reads should be specified in rfi.

If the device uses xon/xoff handshaking for flow control, give xon. Padding information should still be included so that routines can make better decisions about costs, but actual pad characters will not be transmitted. Sequences to turn on and off xon/xoff handshaking may be given in smxon and rmxon. If the characters used for handshaking are not ^S and ^Q, they may be specified with xonc and xoffc.

If the terminal has a "meta key" which acts as a shift key, setting the 8th bit of any character transmitted, this fact can be indicated with km. Otherwise, software will assume that the 8th bit is parity and it will usually be cleared. If strings exist to turn this meta mode on and off, they can be given as smm and rmm.

If the terminal has more lines of memory than will fit on the screen at once, the number of lines of memory can be indicated with lm. A value of lm#0 indicates that the number of lines is not fixed, but that there is still more memory than fits on the screen.

Media copy strings which control an auxiliary printer connected to the terminal can be given as mc0: print the contents of the screen, mc4: turn off the printer, and mc5: turn on the printer. When the printer is on, all text sent to the terminal will be sent to the printer. A variation, mc5p, takes one parameter, and leaves the printer on for as many characters as the value of the parameter, then turns the printer off. The parameter should not exceed 255. If the text is not displayed on the terminal screen when the printer is on, specify mc5i (silent printer). All text, including mc4, is transparently passed to the printer while an mc5p is in effect.

Section 1-15: Special Cases

The working model used by terminfo fits most terminals reasonably well. However, some terminals do not completely match that model, requiring special support by terminfo. These are not meant to be construed as deficiencies in the terminals; they are just differences between the working model and the actual hardware.

They may be unusual devices or, for some reason, do not have all the features of the `terminfo` model implemented.

Terminals that cannot display tilde (~) characters, such as certain Hazeltine terminals, should indicate `hz`.

Terminals that ignore a linefeed immediately after an `am` wrap, such as the Concept 100, should indicate `xenl`. Those terminals whose cursor remains on the right-most column until another character has been received, rather than wrapping immediately upon receiving the right-most character, such as the VT100, should also indicate `xenl`.

If `el` is required to get rid of standout (instead of writing normal text on top of it), `xhp` should be given.

Those Teleray terminals whose tabs turn all characters moved over to blanks, should indicate `xt` (destructive tabs). This capability is also taken to mean that it is not possible to position the cursor on top of a magic cookie. Therefore, to erase standout mode, it is necessary, instead, to use delete and insert line.

Those Beehive Superbee terminals which do not transmit the escape or <CTRL-C> characters, should specify `xsb`, indicating that the f1 key is to be used for escape and the `f2` key for <CTRL-C>.

Section 1-16: Similar Terminals

If there are two very similar terminals, one can be defined as being just like the other with certain exceptions. The string capability `use` can be given with the name of the similar terminal. The capabilities given before `use` override those in the terminal type invoked by `use`. A capability can be canceled by placing *xx@* to the left of the capability definition, where *xx* is the capability. For example, the entry

```
att4424-2|Teletype4424 in display function group ii,□
   rev@, sgr@, smul@, use=att4424,
```

defines an AT&T4424 terminal that does not have the `rev`, `sgr`, and `smul` capabilities, and hence cannot do highlighting. This is useful for different modes for a terminal, or for different user preferences. More than one `use` capability may be given.

PART 2: PRINTER CAPABILITIES

The `terminfo` database allows you to define capabilities of printers as well as terminals. To find out what capabilities are available for printers as well as for terminals, see the two lists under **DEVICE CAPABILITIES** that list capabilities by variable and by capability name.

Section 2-1: Rounding Values

Because parameterized string capabilities work only with integer values, we recommend that `terminfo` designers create strings that expect numeric values that have been rounded. Application designers should note this and should always round values to the nearest integer before using them with a parameterized string capability.

Section 2-2: Printer Resolution

A printer's resolution is defined to be the smallest spacing of characters it can achieve. In general printers have independent resolution horizontally and vertically. Thus the vertical resolution of a printer can be determined by measuring the smallest achievable distance between consecutive printing baselines, while the horizontal resolution can be determined by measuring the smallest achievable distance between the left-most edges of consecutive printed, identical, characters.

All printers are assumed to be capable of printing with a uniform horizontal and vertical resolution. The view of printing that `terminfo` currently presents is one of printing inside a uniform matrix: All characters are printed at fixed positions relative to each cell in the matrix; furthermore, each cell has the same size given by the smallest horizontal and vertical step sizes dictated by the resolution. (The cell size can be changed as will be seen later.)

Many printers are capable of proportional printing, where the horizontal spacing depends on the size of the character last printed. `terminfo` does not make use of this capability, although it does provide enough capability definitions to allow an application to simulate proportional printing.

A printer must not only be able to print characters as close together as the horizontal and vertical resolutions

suggest, but also of moving to a position an integral multiple of the smallest distance away from a previous position. Thus printed characters can be spaced apart a distance that is an integral multiple of the smallest distance, up to the length or width of a single page.

Some printers can have different resolutions depending on different "modes". In normal mode, the existing `terminfo` capabilities are assumed to work on columns and lines, just like a video terminal. Thus the old `lines` capability would give the length of a page in lines, and the `cols` capability would give the width of a page in columns. In micro mode, many `terminfo` capabilities work on increments of lines and columns. With some printers the micro mode may be concomitant with normal mode, so that all the capabilities work at the same time.

Section 2-3: Specifying Printer Resolution

The printing resolution of a printer is given in several ways. Each specifies the resolution as the number of smallest steps per distance:

Characteristic Number of Smallest Steps

`orhi`	Steps per inch horizontally
`orvi`	Steps per inch vertically
`orc`	Steps per column
`orl`	Steps per line

When printing in normal mode, each character printed causes movement to the next column, except in special cases described later; the distance moved is the same as the per-column resolution. Some printers cause an automatic movement to the next line when a character is printed in the rightmost position; the distance moved vertically is the same as the per-line resolution. When printing in micro mode, these distances can be different, and may be zero for some printers.

Automatic Motion after Printing

Normal Mode:	
`orc`	Steps moved horizontally
`orl`	Steps moved vertically
Micro Mode:	
`mcs`	Steps moved horizontally
`mls`	Steps moved vertically

Some printers are capable of printing wide characters. The distance moved when a wide character is printed in normal mode may be different from when a regular width character is printed. The distance moved when a wide character is printed in micro mode may also be different from when a regular character is printed in micro mode, but the differences are assumed to be related: If the distance moved for a regular character is the same whether in normal mode or micro mode (`mcs`=`orc`), then the distance moved for a wide character is also the same whether in normal mode or micro mode. This doesn't mean the normal character distance is necessarily the same as the wide character distance, just that the distances don't change with a change in normal to micro mode. However, if the distance moved for a regular character is different in micro mode from the distance moved in normal mode (`mcs`<`orc`), the micro mode distance is assumed to be the same for a wide character printed in micro mode, as the table below shows.

Automatic Motion after Printing Wide Character

Normal Mode or *Micro Mode* (`mcs` = `orc`):

`widcs` Steps moved horizontally

Micro Mode (`mcs` < `orc`):

`mcs` Steps moved horizontally

There may be control sequences to change the number of columns per inch (the character pitch) and to change the number of lines per inch (the line pitch). If these are used, the resolution of the printer changes, but the type of change depends on the printer:

Changing the Character/Line Pitches

`cpi` Change character pitch

`cpix` If set, `cpi` changes `orhi`, otherwise changes `orc`

`lpi` Change line pitch

`lpix` If set, `lpi` changes `orvi`, otherwise changes `orl`

`chr` Change steps per column

`cvr` Change steps per line

The `cpi` and `lpi` string capabilities are each used with a single argument, the pitch in columns (or characters) and lines per inch, respectively. The `chr` and `cvr` string capabilities are each used with a single argument, the number of steps per column and line, respectively.

Using any of the control sequences in these strings will imply a change in some of the values of `orc`, `orhi`, `orl`, and `orvi`. Also, the distance moved when a wide character is printed, `widcs`, changes in relation to `orc`. The distance moved when a character is printed in micro mode, `mcs`, changes similarly, with one exception: if the distance is 0 or 1, then no change is assumed.

Programs that use `cpi`, `lpi`, `chr`, or `cvr` should recalculate the printer resolution (and should recalculate other values [see below 2-7: "Effect of Changing Printing Resolution" and 2-6: "Dot-Matrix Graphics"]). .

Effects of Changing the Character/Line Pitches

Before	After
Using `cpi` with `cpix` clear:	
$bold orhi ""$	$bold orhi$
$bold orc ""$	$bold orc = bold orhi over V sub italic cpi$
Using `cpi` with `cpix` set:	
$bold orhi ""$	$bold orhi = bold orc cdot V sub italic cpi$
$bold orc ""$	$bold orc$
Using `lpi` with `lpix` clear:	
$bold orvi ""$	$bold orvi$
$bold orl ""$	$bold orl = bold orvi over V sub italic lpi$
Using `lpi` with `lpix` set:	
$bold orvi ""$	$bold orvi = bold orl cdot V sub italic lpi$
$bold orl ""$	$bold orl$
Using `chr`:	
$bold orhi ""$	$bold orhi$
$bold orc ""$	$V sub italic chr$

Using `cvr`:	
$bold orvi ""$	$bold orvi$
$bold orl ""$	$V sub italic cvr$
Using `cpi` or `chr`:	
$bold widcs ""$	$bold widcs = bold {widcs ""} bold orc over { bold {orc ""} }$
$bold mcs ""$	$bold mcs = bold {mcs ""} bold orc over { bold {orc ""} }$

$V sub italic cpi$, $V sub italic lpi$, $V sub italic chr$, and $V sub italic cvr$ are the arguments used with `cpi`, `lpi`, `chr`, and `cvr`, respectively. The prime marks (') indicate the old values. .

Section 2-4: Capabilities that Cause Movement

In the following descriptions, "movement" refers to the motion of the "current position". With video terminals this would be the cursor; with some printers this is the carriage position. Other printers have different equivalents. In general, the current position is where a character would be displayed if printed.

`terminfo` has string capabilities for control sequences that cause movement a number of full columns or lines. It also has equivalent string capabilities for control sequences that cause movement a number of smallest steps.

String Capabilities for Motion

`mcub1`	Move 1 step left
`mcuf1`	Move 1 step right
`mcuu1`	Move 1 step up
`mcud1`	Move 1 step down
`mcub`	Move *N* steps left
`mcuf`	Move *N* steps right
`mcuu`	Move *N* steps up
`mcud`	Move *N* steps down
`mhpa`	Move *N* steps from the left
`mvpa`	Move *N* steps from the top

The latter six strings are each used with a single argument, *N*.

Sometimes the motion is limited to less than the width or length of a page. Also, some printers don't accept absolute motion to the left of the current position. `terminfo` has capabilities for specifying these limits.

Limits to Motion

`mjump`	Limit on use of `mcub1`, `mcuf1`, `mcuu1`, `mcud1`
`maddr`	Limit on use of `mhpa`, `mvpa`
`xhpa`	If set, `hpa` and `mhpa` can't move left
`xvpa`	If set, `vpa` and `mvpa` can't move up

If a printer needs to be in a micro mode for the motion capabilities described above to work, there are string capabilities defined to contain the control sequence to enter and exit this mode. A boolean is available for those printers where using a carriage return causes an automatic return to normal mode.

Entering/Exiting Micro Mode

`smicm`	Enter micro mode

rmicm	Exit micro mode
crxm	Using cr exits micro mode

The movement made when a character is printed in the rightmost position varies among printers. Some make no movement, some move to the beginning of the next line, others move to the beginning of the same line. terminfo has boolean capabilities for describing all three cases.

What Happens After Character Printed in Rightmost Position

sam	Automatic move to beginning of same line

Some printers can be put in a mode where the normal direction of motion is reversed. This mode can be especially useful when there are no capabilities for leftward or upward motion, because those capabilities can be built from the motion reversal capability and the rightward or downward motion capabilities. It is best to leave it up to an application to build the leftward or upward capabilities, though, and not enter them in the terminfo database. This allows several reverse motions to be strung together without intervening wasted steps that leave and reenter reverse mode.

Entering/Exiting Reverse Modes

slm	Reverse sense of horizontal motions
rlm	Restore sense of horizontal motions
sum	Reverse sense of vertical motions
rum	Restore sense of vertical motions
While sense of horizontal motions reversed:	
mcub1	Move 1 step right
mcuf1	Move 1 step left
mcub	Move *N* steps right
mcuf	Move *N* steps left
cub1	Move 1 column right
cuf1	Move 1 column left
cub	Move *N* columns right
cuf	Move *N* columns left
While sense of vertical motions reversed:	
mcuu1	Move 1 step down
mcud1	Move 1 step up
mcuu	Move *N* steps down
mcud	Move *N* steps up
cuu1	Move 1 line down
cud1	Move 1 line up
cuu	Move *N* lines down
cud	Move *N* lines up

The reverse motion modes should not affect the mvpa and mhpa absolute motion capabilities. The reverse vertical motion mode should, however, also reverse the action of the line wrapping that occurs when a character is printed in the right-most position. Thus printers that have the standard terminfo capability am

defined should experience motion to the beginning of the previous line when a character is printed in the right-most position under reverse vertical motion mode.

The action when any other motion capabilities are used in reverse motion modes is not defined; thus, programs must exit reverse motion modes before using other motion capabilities.

Two miscellaneous capabilities complete the list of new motion capabilities. One of these is needed for printers that move the current position to the beginning of a line when certain control characters, such as line-feed or form-feed, are used. The other is used for the capability of suspending the motion that normally occurs after printing a character.

Miscellaneous Motion Strings

`docr`	List of control characters causing `cr`
`zerom`	Prevent auto motion after printing next single character

Margins

`terminfo` provides two strings for setting margins on terminals: one for the left and one for the right margin. Printers, however, have two additional margins, for the top and bottom margins of each page. Furthermore, some printers require not using motion strings to move the current position to a margin and then fixing the margin there, but require the specification of where a margin should be regardless of the current position. Therefore `terminfo` offers six additional strings for defining margins with printers.

Setting Margins

`smgl`	Set left margin at current column
`smgr`	Set right margin at current column
`smgb`	Set bottom margin at current line
`smgt`	Set top margin at current line
`smgbp`	Set bottom margin at line N
`smglp`	Set left margin at column N
`smgrp`	Set right margin at column N
`smgtp`	Set top margin at line N

The last four strings are used with one or more arguments that give the position of the margin or margins to set. If both of `smglp` and `smgrp` are set, each is used with a single argument, N, that gives the column number of the left and right margin, respectively. If both of `smgtp` and `smgbp` are set, each is used to set the top and bottom margin, respectively: `smgtp` is used with a single argument, N, the line number of the top margin; however, `smgbp` is used with two arguments, N and M, that give the line number of the bottom margin, the first counting from the top of the page and the second counting from the bottom. This accommodates the two styles of specifying the bottom margin in different manufacturers' printers. When coding a `terminfo` entry for a printer that has a settable bottom margin, only the first or second parameter should be used, depending on the printer. When writing an application that uses `smgbp` to set the bottom margin, both arguments must be given.

If only one of `smglp` and `smgrp` is set, then it is used with two arguments, the column number of the left and right margins, in that order. Likewise, if only one of `smgtp` and `smgbp` is set, then it is used with two arguments that give the top and bottom margins, in that order, counting from the top of the page. Thus when coding a `terminfo` entry for a printer that requires setting both left and right or top and bottom margins simultaneously, only one of `smglp` and `smgrp` or `smgtp` and `smgbp` should be defined; the other should be left blank. When writing an application that uses these string capabilities, the pairs should be first checked to see if each in the pair is set or only one is set, and should then be used accordingly.

In counting lines or columns, line zero is the top line and column zero is the left-most column. A zero value for

the second argument with `smgbp` means the bottom line of the page.

All margins can be cleared with `mgc`.

Shadows, Italics, Wide Characters, Superscripts, Subscripts

Five new sets of strings are used to describe the capabilities printers have of enhancing printed text.

Enhanced Printing	
`sshm`	Enter shadow-printing mode
`rshm`	Exit shadow-printing mode
`sitm`	Enter italicizing mode
`ritm`	Exit italicizing mode
`swidm`	Enter wide character mode
`rwidm`	Exit wide character mode
`ssupm`	Enter superscript mode
`rsupm`	Exit superscript mode
`supcs`	List of characters available as superscripts
`ssubm`	Enter subscript mode
`rsubm`	Exit subscript mode
`subcs`	List of characters available as subscripts

If a printer requires the `sshm` control sequence before every character to be shadow-printed, the `rshm` string is left blank. Thus programs that find a control sequence in `sshm` but none in `rshm` should use the `sshm` control sequence before every character to be shadow-printed; otherwise, the `sshm` control sequence should be used once before the set of characters to be shadow-printed, followed by `rshm`. The same is also true of each of the `sitm`/`ritm`, `swidm`/`rwidm`, `ssupm`/`rsupm`, and `ssubm`/`rsubm` pairs.

Note that `terminfo` also has a capability for printing emboldened text (`bold`). While shadow printing and emboldened printing are similar in that they "darken" the text, many printers produce these two types of print in slightly different ways. Generally, emboldened printing is done by overstriking the same character one or more times. Shadow printing likewise usually involves overstriking, but with a slight movement up and/or to the side so that the character is "fatter".

It is assumed that enhanced printing modes are independent modes, so that it would be possible, for instance, to shadow print italicized subscripts.

As mentioned earlier, the amount of motion automatically made after printing a wide character should be given in `widcs`.

If only a subset of the printable ASCII characters can be printed as superscripts or subscripts, they should be listed in `supcs` or `subcs` strings, respectively. If the `ssupm` or `ssubm` strings contain control sequences, but the corresponding `supcs` or `subcs` strings are empty, it is assumed that all printable ASCII characters are available as superscripts or subscripts.

Automatic motion made after printing a superscript or subscript is assumed to be the same as for regular characters. Thus, for example, printing any of the following three examples will result in equivalent motion:

`Bi` B_i B^i

Note that the existing `msgr` boolean capability describes whether motion control sequences can be used while in standout mode. This capability is extended to cover the enhanced printing modes added here. `msgr` should be set for those printers that accept any motion control sequences without affecting shadow, italicized, widened, superscript, or subscript printing. Conversely, if `msgr` is not set, a program should end these modes before attempting any motion.

Section 2-5: Alternate Character Sets

In addition to allowing you to define line graphics (described in section 1-12), `terminfo` lets you define alternate character sets. The following capabilities cover printers and terminals with multiple selectable or definable character sets.

Alternate Character Sets

`scs`	Select character set *N*
`scsd`	Start definition of character set *N*, *M* characters
`defc`	Define character *A*, *B* dots wide, descender *D*
`rcsd`	End definition of character set *N*
`csnm`	List of character set names
`daisy`	Printer has manually changed print-wheels

The `scs`, `rcsd`, and `csnm` strings are used with a single argument, *N*, a number from 0 to 63 that identifies the character set. The `scsd` string is also used with the argument *N* and another, *M*, that gives the number of characters in the set. The `defc` string is used with three arguments: *A* gives the ASCII code representation for the character, *B* gives the width of the character in dots, and *D* is zero or one depending on whether the character is a "descender" or not. The `defc` string is also followed by a string of image-data bytes that describe how the character looks (see below).

Character set 0 is the default character set present after the printer has been initialized. Not every printer has 64 character sets, of course; using `scs` with an argument that doesn't select an available character set should cause a null result from `tparm`.

If a character set has to be defined before it can be used, the `scsd` control sequence is to be used before defining the character set, and the `rcsd` is to be used after. They should also cause a null result from `tparm` when used with an argument *N* that doesn't apply. If a character set still has to be selected after being defined, the `scs` control sequence should follow the `rcsd` control sequence. By examining the results of using each of the `scs`, `scsd`, and `rcsd` strings with a character set number in a call to `tparm`, a program can determine which of the three are needed.

Between use of the `scsd` and `rcsd` strings, the `defc` string should be used to define each character. To print any character on printers covered by `terminfo`, the ASCII code is sent to the printer. This is true for characters in an alternate set as well as normal characters. Thus the definition of a character includes the ASCII code that represents it. In addition, the width of the character in dots is given, along with an indication of whether the character should descend below the print line (such as the lowercase letter g in most character sets). The width of the character in dots also indicates the number of image-data bytes that will follow the `defc` string. These image-data bytes indicate where in a dot-matrix pattern ink should be applied to "draw" the character; the number of these bytes and their form are defined below under 2-6: "Dot-Matrix Graphics".

It's easiest for the creator of `terminfo` entries to refer to each character set by number; however, these numbers will be meaningless to the application developer. The `csnm` string alleviates this problem by providing names for each number.

When used with a character set number in a call to `tparm`, the `csnm` string will produce the equivalent name. These names should be used as a reference only. No naming convention is implied, although anyone who creates a `terminfo` entry for a printer should use names consistent with the names found in user documents for the printer. Application developers should allow a user to specify a character set by number (leaving it up to the user to examine the `csnm` string to determine the correct number), or by name, where the application examines the `csnm` string to determine the corresponding character set number.

These capabilities are likely to be used only with dot-matrix printers. If they are not available, the strings should not be defined. For printers that have manually changed print-wheels or font cartridges, the boolean `daisy` is set.

Section 2-6: Dot-Matrix Graphics

Dot-matrix printers typically have the capability of reproducing raster-graphics images. Three new numeric capabilities and three new string capabilities can help a program draw raster-graphics images independent of the type of dot-matrix printer or the number of pins or dots the printer can handle at one time.

Dot-Matrix Graphics

`npins`	Number of pins, *N*, in print-head
`spinv`	Spacing of pins vertically in pins per inch
`spinh`	Spacing of dots horizontally in dots per inch
`porder`	Matches software bits to print-head pins
`sbim`	Start printing bit image graphics, *B* bits wide
`rbim`	End printing bit image graphics

The `sbim` sring is used with a single argument, *B*, the width of the image in dots.

The model of dot-matrix or raster-graphics that `terminfo` presents is similar to the technique used for most dot-matrix printers: each pass of the printer's print-head is assumed to produce a dot-matrix that is *N* dots high and *B* dots wide. This is typically a wide, squat, rectangle of dots. The height of this rectangle in dots will vary from one printer to the next; this is given in the `npins` numeric capability. The size of the rectangle in fractions of an inch will also vary; it can be deduced from the `spinv` and `spinh` numeric capabilities. With these three values an application can divide a complete raster-graphics image into several horizontal strips, perhaps interpolating to account for different dot spacing vertically and horizontally.

The `sbim` and `rbim` strings are used to start and end a dot-matrix image, respectively. The `sbim` string is used with a single argument that gives the width of the dot-matrix in dots. A sequence of image-data bytes are sent to the printer after the `sbim` string and before the `rbim` string. The number of bytes is a integral multiple of the width of the dot-matrix; the multiple and the form of each byte is determined by the `porder` string as described below.

The `porder` string is a comma separated list of pin numbers optionally followed by an numerical offset. The offset, if given, is separated from the list with a semicolon. The position of each pin number in the list corresponds to a bit in an 8-bit data byte. The pins are numbered consecutively from 1 to `npins`, with 1 being the top pin. Note that the term "pin" is used loosely here; ink-jet dot-matrix printers don't have pins, but can be considered to have an equivalent method of applying a single dot of ink to paper. The bit positions in `porder` are in groups of 8, with the first position in each group the most significant bit and the last position the least significant bit. An application produces 8-bit bytes in the order of the groups in `porder`.

An application computes the image-data bytes from the internal image, mapping vertical dot positions in each print-head pass into 8-bit bytes, using a 1 bit where ink should be applied and 0 where no ink should be applied. This can be reversed (0 bit for ink, 1 bit for no ink) by giving a negative pin number. If a position is skipped in `porder`, a 0 bit is used. If a position has a lowercase x instead of a pin number, a 1 bit is used in the skipped position. For consistency, a lowercase o can be used to represent a 0 filled, skipped bit. There must be a multiple of 8 bit positions used or skipped in `porder`; if not, 0 bits are used to fill the last byte in the least significant bits. The offset, if given, is added to each data byte; the offset can be negative.

Some examples may help clarify the use of the `porder` string. The AT&T 470, AT&T 475 and C.Itoh 8510 printers provide eight pins for graphics. The pins are identified top to bottom by the 8 bits in a byte, from least significant to most. The `porder` strings for these printers would be `8,7,6,5,4,3,2,1`. The AT&T 478 and AT&T 479 printers also provide eight pins for graphics. However, the pins are identified in the reverse order. The `porder` strings for these printers would be `1,2,3,4,5,6,7,8`. The AT&T 5310, AT&T 5320, DEC LA100, and DEC LN03 printers provide six pins for graphics. The pins are identified top to bottom by the decimal values 1, 2, 4, 8, 16 and 32. These correspond to the low six bits in an 8-bit byte, although the decimal values are further offset by the value 63. The `porder` string for these printers would be `,,6,5,4,3,2,1;63`, or alternately `o,o,6,5,4,3,2,1;63`.

Section 2-7: Effect of Changing Printing Resolution

If the control sequences to change the character pitch or the line pitch are used, the pin or dot spacing may change:

Changing the Character/Line Pitches

`cpi`	Change character pitch
`cpix`	If set, `cpi` changes `spinh`
`lpi`	Change line pitch
`lpix`	If set, `lpi` changes `spinv`

Programs that use `cpi` or `lpi` should recalculate the dot spacing: .

Effects of Changing the Character/Line Pitches

Before	After
Using `cpi` with `cpix` clear:	
$bold spinh """$	$bold spinh$
Using `cpi` with `cpix` set:	
$bold spinh """$	$bold spinh = bold spinh """ cdot bold orhi over { bold {orhi """} } $
Using `lpi` with `lpix` clear:	
$bold spinv """$	$bold spinv$
Using `lpi` with `lpix` set:	
$bold spinv """$	$bold spinv = bold {spinv """} cdot bold orhi over { bold {orhi """}} $
Using `chr:`	
$bold spinh """$	$bold spinh$
Using `cvr:`	
$bold spinv """$	$bold spinv$

orhi' and **orhi** are the values of the horizontal resolution in steps per inch, before using `cpi` and after using `cpi`, respectively. Likewise, **orvi'** and **orvi** are the values of the vertical resolution in steps per inch, before using `lpi` and after using `lpi`, respectively. Thus, the changes in the dots per inch for dot-matrix graphics follow the changes in steps per inch for printer resolution. .

Section 2-8: Print Quality

Many dot-matrix printers can alter the dot spacing of printed text to produce near letter quality printing or draft quality printing. Usually it is important to be able to choose one or the other because the rate of printing generally falls off as the quality improves. There are three new strings used to describe these capabilities.

Print Quality

`snlq`	Set near-letter quality print
`snrmq`	Set normal quality print
`sdrfq`	Set draft quality print

The capabilities are listed in decreasing levels of quality. If a printer doesn't have all three levels, one or two of

the strings should be left blank as appropriate.

Section 2-9: Printing Rate and Buffer Size

Because there is no standard protocol that can be used to keep a program synchronized with a printer, and because modern printers can buffer data before printing it, a program generally cannot determine at any time what has been printed. Two new numeric capabilities can help a program estimate what has been printed.

Print Rate/Buffer Size	
`cps`	Nominal print rate in characters per second
`bufsz`	Buffer capacity in characters

`cps` is the nominal or average rate at which the printer prints characters; if this value is not given, the rate should be estimated at one-tenth the prevailing baud rate. `bufsz` is the maximum number of subsequent characters buffered before the guaranteed printing of an earlier character, assuming proper flow control has been used. If this value is not given it is assumed that the printer does not buffer characters, but prints them as they are received.

As an example, if a printer has a 1000-character buffer, then sending the letter "a" followed by 1000 additional characters is guaranteed to cause the letter "a" to print. If the same printer prints at the rate of 100 characters per second, then it should take 10 seconds to print all the characters in the buffer, less if the buffer is not full. By keeping track of the characters sent to a printer, and knowing the print rate and buffer size, a program can synchronize itself with the printer.

Note that most printer manufacturers advertise the maximum print rate, not the nominal print rate. A good way to get a value to put in for `cps` is to generate a few pages of text, count the number of printable characters, and then see how long it takes to print the text.

Applications that use these values should recognize the variability in the print rate. Straight text, in short lines, with no embedded control sequences will probably print at close to the advertised print rate and probably faster than the rate in `cps`. Graphics data with a lot of control sequences, or very long lines of text, will print at well below the advertised rate and below the rate in `cps`. If the application is using `cps` to decide how long it should take a printer to print a block of text, the application should pad the estimate. If the application is using `cps` to decide how much text has already been printed, it should shrink the estimate. The application will thus err in favor of the user, who wants, above all, to see all the output in its correct place.

Section 2-10: Selecting a Terminal

If the environment variable `TERMINFO` is defined, any program using `curses` checks for a local terminal definition before checking in the standard place. For example, if `TERM` is set to `att4424`, then the compiled terminal definition is found in by default the path:

`a/att4424`

within an implementation-specific directory.

(The `a` is copied from the first letter of `att4424` to avoid creation of huge directories.) However, if `TERMINFO` is set to `$HOME/myterms`, `curses` first checks:

`$HOME/myterms/a/att4424`

If that fails, it then checks the default pathname.

This is useful for developing experimental definitions or when write permission in the implementation-defined default database is not available.

If the `LINES` and `COLUMNS` environment variables are set, or if the program is executing in a window environment, line and column information in the environment will override information read by `terminfo`.

NOTES

The most effective way to prepare a terminal description is by imitating the description of a similar terminal in `terminfo` and to build up a description gradually, using partial descriptions with a screen oriented editor,

such as `vi`, to check that they are correct. To easily test a new terminal description the environment variable `TERMINFO` can be set to the pathname of a directory containing the compiled description, and programs will look there rather than in `/usr/share/lib/terminfo`.

It is implementation-defined how the entries in `terminfo` may be created.

There is more than one way to write a `terminfo` entry. A minimal entry may permit applications to use `curses` to operate the terminal. If the entry is enhanced to describe more of the terminal's capabilities, applications can use `curses` to invoke those features, and can take advantages of optimizations within `curses` and thus operate more efficiently. For most terminals, an optimal `terminfo` entry has already been written.

FILES

`/usr/share/lib/terminfo/?/*`
: Compiled terminal description database

`/usr/share/lib/tabset/*`
: Tab settings for some terminals, in a format appropriate to be output to the terminal (escape sequences that set margins and tabs)

SEE ALSO

`stty(1)`, `tput(1)`, `tty(1)`, `tic(1M)`, `curses(3X)`, `term(4)`, `curses(5)`, `term(5)`, `term_names(5)`.

System Administrator's Reference Manual

termtab(4)

NAME

`termtab` - device database

SYNOPSIS

`/var/sadm/terms/termtab`

DESCRIPTION

The `termtab` file contains information on the attributes of devices which are connected via TACLAN, MUX, or DPTG. This file is processed using `termadm`. It contains the information required to start and end the daemon processes for managing a TACLAN device. A minimum of two lines are therefore available with the following information:

tag:*hostname*:*terminalname*:*portno*:*termtype*:*admin-channel*:*second*:*command*

The individual entries are separated from one another by a colon (:). Comment lines begin with a number character (#) in column 1.

`termadd` generates two lines per device configured: one with the tag column "start" together with a command for starting the relevant daemon process which establishes an Ethernet connection to an entry in `/dev/term`, and the other line with the tab column "stop" containing a `kill` command for ending this daemon process. The device name cannot be a path name; i.e. the device name must be specified without a directory name. Only basic file names are permitted.

`termadm` searches through the file and selects a line if the specified field matches a value in the relevant column [see `termadm`(1M)].

When configuring a device, `termadd` sets every column to a suitable value. To speed up processing, `termadm` supports the stop operation for every specified device. It calls the process number of the relevant daemon process and stores it in the `PID` variable. This variable is exported and can be used in the command which is used to stop this device.

NOTES

Two-line or multi-line commands are not permitted.

The tag field contains the value "backup" instead of the value "start" for `backup` configurations. Thus, `backup` connections must be started and ended manually.

The `termtab` file can also be extended by a user. However, you must always ensure that this table is used by `termadd`, `termdel`, `termshow`, and `termadm` to manage the device configuration on a specific machine. An error in the table can result in this command malfunctioning. It is particularly important that the lines for starting and stopping the device are not changed.

EXAMPLE

```
start:myannex:myterm:1:ba80:myannex@1::/opt/bin/dptg2/tsvr_d -a ...
stop:myannex:myterm:1:ba80:myannex@1::KILLTSVR_D
start:myannex:anyterm:0:97801-vt:myannex@0::rtelnet ...
stop:myannex:anyterm:0:97801-vt:myannex@0::KILLRTELNET
backup:newannex:failsafe:0:97801-5::/usr/bin/rtelnet ...
stop:newannex:failsafe:0:97801-5::KILLRTELNET
start::kh:3:9766:/dev/dptg/itp0.3::dptg_lk -a ...
stop::kh:3:9766:/dev/dptg/itp0.3::UNLINKDPTG
```

SEE ALSO

`termadd`(1M) , `termadm`(1M) , `termdel`(1M) .

System Administrator's Reference Manual

timezone(4)

NAME

`timezone -` set default system time zone

SYNOPSIS

`/etc/TIMEZONE`

DESCRIPTION

This file sets and exports the time zone environment variable `TZ`.

This file is "dotted" into other files that must know the time zone.

EXAMPLES

`/etc/TIMEZONE` for the east coast:

```
# Time Zone
TZ=EST5EDT
export TZ
```

`/etc/TIMEZONE` for Central Europe:

```
# Time Zone
TZ=MEZ-1MSZ-2
export TZ
```

SEE ALSO

`rc2`(1M), `ctime`(3C), `profile`(4), `environ`(5).

System Administrator's Reference Manual

ts_dptbl(4)

NAME

`ts_dptbl` - time-sharing dispatcher parameter table

DESCRIPTION

The process scheduler (or dispatcher) is the portion of the kernel that controls allocation of the CPU to processes. The scheduler supports the notion of scheduling classes where each class defines a scheduling policy, used to schedule processes within that class. Associated with each scheduling class is a set of priority queues on which ready to run processes are linked. These priority queues are mapped by the system configuration into a set of global scheduling priorities which are available to processes within the class. (The dispatcher always selects for execution the process with the highest global scheduling priority that is eligible to run on a given CPU. See **PROCESSOR AFFINITY AND BINDING CONSIDERATIONS** below.) The priority queues associated with a given class are viewed by that class as a contiguous set of priority levels numbered from 0 (lowest priority) to *n* (highest priority – a configuration-dependent value). The set of global scheduling priorities that the queues for a given class are mapped into might not start at zero and might not be contiguous (depending on the configuration).

Processes in the time-sharing class which are running in user mode (or in kernel mode before going to sleep) are scheduled according to the parameters in a time-sharing dispatcher parameter table (`ts_dptbl`). (Time-sharing processes running in kernel mode after sleeping are run within a special range of priorities reserved for such processes and are not affected by the parameters in the `ts_dptbl` until they return to user mode.) The `ts_dptbl` consists of an array of parameter structures (`struct ts_dpent`), one for each of the *n* priority levels used by time-sharing processes in user mode. The properties of a given priority level *i* are specified by the *i*th parameter structure in this array (`ts_dptbl`[*i*]).

A parameter structure consists of the following members. These are also described in the `/usr/include/sys/ts.h` header file.

`ts_globpri`
: The global scheduling priority associated with this priority level. The mapping between time-sharing priority levels and global scheduling priorities is determined at boot time by the system configuration. `ts_globpri` is the only member of the `ts_dptbl` which cannot be changed with `dispadmin`(1M) .

`ts_quantum`
: The length of the time quantum allocated to processes at this level in ticks (HZ).

`ts_tqexp`
: Priority level of the new queue on which to place a process running at the current level if it exceeds its time quantum. Normally this field links to a lower priority time-sharing level that has a larger quantum.

`ts_slpret`
: Priority level of the new queue on which to place a process, that was previously in user mode at this level, when it returns to user mode after sleeping. Normally this field links to a higher priority level that has a smaller quantum.

`ts_maxwait`
: A per process counter, `ts_dispwait` is initialized to zero each time a time-sharing process is placed back on the dispatcher queue after its time quantum has expired or when it is awakened (`ts_dispwait` is not reset to zero when a process is preempted by a higher priority process). This counter is incremented once per second for each process on the dispatcher queue. If a process's `ts_dispwait` value exceeds the `ts_maxwait` value for its level, the process's priority is changed to that indicated by `ts_lwait`. The purpose of this field is to prevent starvation.

`ts_lwait`
: Move a process to this new priority level if `ts_dispwait` is greater than `ts_maxwait`.

An administrator can affect the behavior of the time-sharing portion of the scheduler by reconfiguring the

`ts_dptbl`. There are two methods available for doing this.

KERNEL LINK TIME CONFIGURATION

The `ts_dptbl` can be reconfigured at kernel link time time by specifying the desired values in the `/etc/conf/pack.d/ts/space.c` file and reconfiguring the system using the auto-configuration procedure [see `idbuild(1M)`]. This is the only method that can be used to change the number of time share priority levels or the set of global scheduling priorities used by the time share class.

DISPADMIN CONFIGURATION FILE

With the exception of `ts_globpri` all of the members of the `ts_dptbl` can be examined and modified on a running system using the `dispadmin(1M)` command. Invoking `dispadmin` for the time-sharing class allows the administrator to retrieve the current `ts_dptbl` configuration from the kernel's in-core table, or overwrite the in-core table with values from a configuration file. The configuration file used for input to `dispadmin` must conform to the specific format described below.

Blank lines are ignored and any part of a line to the right of a `#` symbol is treated as a comment. The first non-blank, non-comment line must indicate the resolution to be used for interpreting the `ts_quantum` time quantum values. The resolution is specified as

`RES=`*res*

where *res* is a positive integer between 1 and 1,000,000,000 inclusive and the resolution used is the reciprocal of *res* in seconds (for example, `RES=1000` specifies millisecond resolution). Although very fine (nanosecond) resolution may be specified, the time quantum lengths are rounded up to the next integral multiple of the system clock's resolution.

The remaining lines in the file are used to specify the parameter values for each of the time-sharing priority levels. The first line specifies the parameters for time-sharing level 0, the second line specifies the parameters for time-sharing level 1, etc. There must be exactly one line for each configured time-sharing priority level.

EXAMPLE

The following excerpt from a `dispadmin` configuration file illustrates the format. Note that for each line specifying a set of parameters there is a comment indicating the corresponding priority level. These level numbers indicate priority within the time-sharing class, and the mapping between these time-sharing priorities and the corresponding global scheduling priorities is determined by the configuration specified in the `ts/space.c` file. The level numbers are strictly for the convenience of the administrator reading the file and, as with any comment, they are ignored by `dispadmin`. `dispadmin` assumes that the lines in the file are ordered by consecutive, increasing priority level (from 0 to the maximum configured time-sharing priority). The level numbers in the comments should normally agree with this ordering; if for some reason they don't, however, `dispadmin` is unaffected.

```
# Time Sharing Dispatcher Configuration
RES=1000
# ts_quantum  ts_tqexp  ts_slpret  ts_maxwait ts_lwait  PRIORITY LEVEL
      1000         0        10           5        10        #     0
      1000         0        11           5        11        #     1
      1000         1        12           5        12        #     2
      1000         1        13           5        13        #     3
      1000         2        14           5        14        #     4
      1000         2        15           5        15        #     5
      1000         3        16           5        16        #     6
      1000         3        17           5        17        #     7
      1000         4        18           5        18        #     8
      1000         4        19           5        19        #     9
       800         5        20           5        20        #    10
       800         5        21           5        21        #    11
         .         .         .           .         .        .     .
       100        48        59           5        59        #    58
       100        49        59           5        59        #    59
```

PROCESSOR AFFINITY AND BINDING CONSIDERATIONS

In order to increase the cache hit rates of a multiprocessor, a process should generally execute on a particular processor. This affinity for a particular processor will reduce the required memory bandwidth for the process (and for the system as a whole). If a process has not run "recently" then it probably only has a small cache state in any given processor – and thus should loose its affinity for any particular processor.

This affinity mechanism can be enabled or disabled on a per-process basis by the super user [see `mpcntl`(1M), `mpcntl`(3X)]. The affinity mechanism defaults to `on` for the `TS` class.

A process can also be bound to a set of processors. If a processor has been so bound, then it will *only* run on one of the processors of that set. If there is a idle processor that is not in the processes binding set – then the process will still wait until there is a processor in its set that is not running a higher priority process.

Process bindings can be established by the process [see `mpcntl`(3X)], or can be temporarily established by the kernel (if the kernel is executing driver code that must run on a particular CPU).

NOTES

`dispadmin` does some limited sanity checking on the values supplied in the configuration file. The sanity checking is intended to ensure that the new `ts_dptbl` values do not cause the system to panic. The sanity checking does not attempt to analyze the effect that the new values will have on the performance of the system. Unusual `ts_dptbl` configurations may have a dramatic negative impact on the performance of the system.

No sanity checking is done on the `ts_dptbl` values specified in the `ts/space.c` file. Specifying an inconsistent or nonsensical `ts_dptbl` configuration through the `ts/space.c` file could cause serious performance problems and/or cause the system to panic.

FILES

`/usr/include/sys/ts.h`
`/etc/conf/pack.d/ts/space.c`

SEE ALSO

`priocntl`(1), `dispadmin`(1M), `idbuild`(1M), `mpcntl`(1M), `priocntl`(2), `mpcntl`(3X).

System Administrator's Reference Manual

ttydefs(4)

NAME

`ttydefs -` file contains terminal line settings information forttymon

SYNOPSIS

`/etc/ttydefs`

DESCRIPTION

`/etc/ttydefs` is an administrative file that contains information used by `ttymon` to set up the speed and terminal settings for a TTY port.

The `ttydefs` file contains the following fields:

ttylabel
: The string `ttymon` tries to match against the TTY port's *ttylabel* field in the port monitor administrative file. It often describes the speed at which the terminal is supposed to run, for example, `1200`.

initial-flags
: Contains the initial `termio(7)` settings to which the terminal is to be set. For example, the system administrator will be able to specify what the default erase and kill characters will be. *initial-flags* must be specified in the syntax recognized by the `stty` command.

final-flags
: *final-flags* must be specified in the same format as *initial-flags*. `ttymon` sets these final settings after a connection request has been made and immediately prior to invoking a port's service.

autobaud
: If the autobaud field contains the character "`A`", autobaud will be enabled. Otherwise, autobaud will be disabled. `ttymon` determines what line speed to set the TTY port to by analyzing the carriage returns entered. If autobaud has been disabled, the hunt sequence is used for baud rate determination.

nextlabel
: If the user indicates that the current terminal setting is not appropriate by sending a BREAK, `ttymon` searches for a `ttydefs` entry whose *ttylabel* field matches the *nextlabel* field. If a match is found, `ttymon` uses that field as its *ttylabel* field. A series of speeds is often linked together in this way into a closed set called a hunt sequence. For example, `4800` may be linked to `1200`, which in turn is linked to `2400`, which is finally linked to `4800`.

SEE ALSO

`ttyadm(1M)`, `ttymon(1M)`, `sttydefs(1M)`.

System Administrator's Reference Manual

ttysrch(4)

NAME

`ttysrch` - directory search list for ttyname

SYNOPSIS

`/etc/ttysrch`

DESCRIPTION

`ttysrch` is an optional file that is used by the `ttyname(3C)` library routine. This file contains the names of files and directories in `/dev` that contain terminal and terminal-related device files. The purpose of this file is to improve the performance of `ttyname` by indicating which files and subdirectories in `/dev` contain terminal-related device files and should be searched first. These file and subdirectory names must appear on separate lines and must begin with `/dev`. Those path names that do not begin with `/dev` will be ignored and a warning will be sent to the console. Blank lines (lines containing only white space) and lines beginning with the comment character "#" will be ignored. For each file listed (except for the special entry `/dev`), `ttyname` will recursively search through subdirectories looking for a match. If `/dev` appears in the `ttysrch` file, the `/dev` directory itself will be searched but there will not be a recursive search through its subdirectories.

When `ttyname` searches through the device files, it tries to find a file whose major/minor device number, file system identifier, and inode number match that of the file descriptor it was given as an argument. If a match is not found, it will settle for a match of just major/minor device and file system identifier, if one can be found. However, if the file descriptor is associated with a cloned device [see `clone(7)`], this algorithm does not work efficiently because the inode number of the device file associated with a clonable device will never match the inode number of the file descriptor that was returned by the open of that clonable device. To help with these situations, entries can be put into the `/etc/ttysrch` file to improve performance when cloned devices are used as terminals on a system (e.g. for remote login). However, this is only useful if the minor devices related to a cloned device are put into a subdirectory. (It is important to note that device files need not exist for cloned devices and if that is the case, `ttyname` will eventually fail.) For example if `/dev/tcp` is a cloned device, there could be a subdirectory `/dev/tcp` that contains files `0`, `1`, `2`, etc. that correspond to the minor devices of the `starlan` driver. An optional second field is used in the `/etc/ttysrch` file to indicate the matching criteria. This field is separated by white space (any combination of blanks or tabs). The letter `M` means major/minor device number, `F` means file system identifier, and `I` means inode number.

The following options (letters) are not standard in UNIX V5.4, but may be used to increase the performance of `ttyname`.

The `f` option provides for rapid access to inodes. It can be used to accelerate searching for an inode, being particularly useful on systems with a large number of terminals or pseudo-tty devices. **Caution**: If the `f` option is set, subdirectories are not searched recursively.

The letter `X` specifies that this particular directory should not be searched for a match.

The letter `N` specifies that a major device number follows as argument. The allows one to specify which major numbers occur in a particular directory. The letter `N` and the following major device number argument may occur more than once in the line, thereby specifying more than major device number for a particular directory. If the major number of the file descriptor passed to `ttyname` matches any of those specified with the `N` option, then `ttyname` searches that directory first looking for a match. This improves the performance of `ttyname` by pointing it to the directory which it should search first.

The letter `P` specifies that a device name pattern follows as argument. This allows one to specify how the names of the files in the directory will be formed for the testing done by `ttyname`.

In order to avoid having to check each file separately in a directory, `ttyname` first attempts to check the filename entry with a matching minor device number to that passed to `ttyname`. In order to do that though, `ttyname` needs to know what the filenames in the directory look like. `ttyname` has default patterns which it uses, but the letter `P` may be used to specify these patterns. The specified pattern must contain `%d` somewhere within it. The `%d` portion is then replaced by the minor device number. As in the `printf` function,

a field width may also be specified, for example, `%3d`. With the `/etc/ttysrch` entry

```
/dev/term P tty%3d
```

and assuming a minor device number of 20, the name argument used by `ttyname` for its testing would be `/dev/term/tty020`.

The letter `E` specifies that the entry is a file and not a directory.

If no option is specified for an entry, the default is `MFI` which means try to match on all three. For cloned devices the field should be `MF`, which indicates that it is not necessary to match on the inode number.

Without the `/etc/ttysrch` file, `ttyname` will search the `/dev` directory by first looking in the directories `/dev/term`, `/dev/pts`, and `/dev/xt`. If a system has terminal devices installed in directories other than these, it may help performance if the `ttysrch` file is created and contains that list of directories.

EXAMPLES

A sample `/etc/ttysrch` file follows:

```
/dev/term/ttyc0 MFE
/dev/inet MFN 29 P %d N 30
/dev/term fIP tty%3d
/dev/pts  MFI
/dev/xt   MFI
/dev/slan MF
```

This file tells `ttyname` that it should first search through those directories listed and that when searching through the `/dev/slan` directory, if a file is encountered whose major/minor devices and file system identifier match that of the file descriptor argument to `ttyname`, this device name should be considered a match.

When `/dev/term` is searched, `ttyname` only compares the inodes. The subdirectories of `/dev/term` are not searched recursively.

SEE ALSO

`ttyname`(3C), `clone`(7).

System Administrator's Reference Manual

unistd(4)

NAME

`unistd -` standard symbolic constants and types

SYNOPSIS

```
#include <unistd.h>
```

DESCRIPTION

The `<unistd.h>` header defines miscellaneous symbolic constants and types, and declares miscellaneous functions. The contents of this header are shown below.

Version Test Macros

The following symbolic constants are defined:

`_POSIX_VERSION`
: Integer value indicating version of the ISO POSIX-1 standard (C language binding).

`_POSIX2_VERSION`
: Integer value indicating version of the ISO POSIX-2 standard (Commands).

`_POSIX2_C_VERSION`
: Integer value indicating version of the ISO POSIX-2 standard (C language binding) and whether the X/Open POSIX2 C-language Binding Feature Group is supported.

`_XOPEN_VERSION`
: Integer value indicating version of the X/Open Portability Guide to which the implementation conforms.

`_POSIX_VERSION` is defined in the ISO POSIX-1 standard. It changes with each new version of the ISO POSIX-1 standard.

`_POSIX2_VERSION` is defined in the ISO POSIX-2 standard. It changes with each new version of the ISO POSIX-2 standard.

`_POSIX2_C_VERSION` is defined in the ISO POSIX-2 standard. It changes with each new version of the ISO POSIX-2 standard. When the C language binding option of the ISO POSIX-2 standard and therefore the X/Open POSIX2 C-language Binding Feature Group is not supported, `_POSIX2_C_VERSION` will be set to −1.

`_XOPEN_VERSION` is defined as an integer value equal to 500.

`_XOPEN_XCU_VERSION` is defined as an integer value indicating the version of the XCU (= X/Open Commands and Utilities) to which the implementation conforms. If the value is −1, no commands and utilities are provided on the implementation. If the value is greater than or equal to 4, the functionality associated with the following symbols is also supported:

```
_POSIX2_C_BIND
_POSIX2_C_VERSION
_POSIX2_CHAR_TERM
_POSIX2_LOCALEDEF
_POSIX2_UPE
_POSIX2_VERSION
```

If this constant is not defined, use the `sysconf()` function to determine which features are supported.

Each of the following symbolic constants is defined only if the implementation supports the indicated revision of the X/Open Portability Guide:

`_XOPEN_XPG2`
: X/Open Portability Guide, Volume 2, January 1987, XVS System Calls and Libraries (ISBN: 0−444−70175−3).

`_XOPEN_XPG3`
: X/Open Specification, February 1992, System Interfaces and Headers, Issue 3 (ISBN: 1−872630−37−5, C212); this specification was formerly X/Open Portability Guide, Issue 3,

Volume 2, January 1989, XSI System Interface and Headers (ISBN:□0−13−685843−0, XO/XPG/89/003).

`_XOPEN_XPG4`
X/Open CAE Specification, July 1992, System Interfaces and Headers, Issue 4 (ISBN:□1−872630−47−2, C202).

`_XOPEN_UNIX`
X/Open CAE Specification, August 1994, System Interfaces and Headers, Issue 4, Version 2 (ISBN: 1-85912-037-7, C435).

Mandatory Symbolic Constants

Although all implementations conforming to this description support all of the FIPS (= Federal Information Processing Standards) features described below, there may be system-dependent or file-system-dependent configuration procedures that can remove or modify any or all of these features. Such configurations should not be made if strict FIPS compliance is required.

The following symbolic constants are either undefined or defined with a value other than −1. If a constant is undefined, an application should use the `sysconf()`, `pathconf()` or `fpathconf()` functions to determine which features are present on the system at that time or for the particular pathname in question.

`_POSIX_CHOWN_RESTRICTED`
The use of `chown()` is restricted to a process with appropriate privileges, and to changing the group ID of a file only to the effective group ID of the process or to one of its supplementary group IDs.

`_POSIX_NO_TRUNC`
Pathname components longer than `NAME_MAX` generate an error.

`_POSIX_VDISABLE`
Terminal special characters defined in `<termios.h>` [see `termios(5)`] can be disabled using this character value.

`_POSIX_SAVED_IDS`
Each process has a saved set-user-ID and a saved set-group-ID.

`_POSIX_JOB_CONTROL`
Implementation supports job control.

`_POSIX_CHOWN_RESTRICTED`, `_POSIX_NO_TRUNC` and `_POSIX_VDISABLE` will have values other than −1. The following symbolic constants are always defined to unspecified values to indicate that this functionality from the POSIX Threads Extension is always present on XSI-conformant systems:

`_POSIX_THREADS`
The implementation supports the threads option.

`_POSIX_THREAD_ATTR_STACKADDR`
The implementation supports the thread stack address attribute option.

`_POSIX_THREAD_ATTR_STACKSIZE`
The implementation supports the thread stack size attribute option.

`_POSIX_THREAD_PROCESS_SHARED`
The implementation supports the process-shared synchronisation option.

`_POSIX_THREAD_SAFE_FUNCTIONS`
The implementation supports the thread-safe functions option.

Constants for Options and Feature Groups

The following symbolic constants are defined to have the value −1 if the implementation will never provide the feature, and to have a value other than −1 if the implementation always provides the feature. If these are undefined, the `sysconf()` function can be used to determine whether the feature is provided for a particular invocation of the application.

`_POSIX2_C_BIND`
Implementation supports the C language binding option. This will always have a value other than −1.

`_POSIX2_C_DEV`

Implementation supports the C language development utilities option.

`_POSIX2_CHAR_TERM`
Implementation supports at least one terminal type.

`_POSIX2_FORT_DEV`
Implementation supports the FORTRAN Development Utilities Option.

`_POSIX2_FORT_RUN`
Implementation supports the FORTRAN Run-time Utilities Option.

`_POSIX2_LOCALEDEF`
Implementation supports the creation of locales by the `localedef` utility.

`_POSIX2_SW_DEV`
Implementation supports the Software Development Utilities Option.

`_POSIX2_UPE`
The implementation supports the User Portability Utilities Option.

`_XOPEN_ENH_I18N`
The implementation supports the Issue 4, Version 2 Enhanced Internationalisation Feature Group. This is always set to a value other than −1.

`_XOPEN_LEGACY`
The implementation supports the Legacy Feature Group.

`_XOPEN_REALTIME_THREADS`
The implementation supports the X/Open Realtime Threads Feature Group.

`_XOPEN_SHM`
The implementation supports the Issue 4, Version 2 Shared Memory Feature Group. This is always set to a value other than −1.

`_XBS5_ILP32_OFF32`
Implementation provides a C-language compilation environment with 32-bit `int`, `long`, `pointer` and `off_t` types.

`_XBS5_ILP32_OFFBIG`
Implementation provides a C-language compilation environment with 32-bit `int`, `long` and `pointer` types and an `off_t` type using at least 64 bits.

`_XBS5_LP64_OFF64`
Implementation provides a C-language compilation environment with 32-bit `int` and 64-bit `long`, `pointer` and `off_t` types.

`_XBS5_LPBIG_OFFBIG`
Implementation provides a C-language compilation environment with an `int` type using at least 32 bits and `long`, `pointer` and `off_t` types using at least 64 bits.

If `_XOPEN_REALTIME_THREADS` is defined to have a value other than −1, then the following symbolic constants will be defined to an unspecified value to indicate that the features are supported:

`_POSIX_THREAD_PRIORITY_SCHEDULING`
The implementation supports the thread execution scheduling option.

`_POSIX_THREAD_PRIO_INHERIT`
The implementation supports the priority inheritance option.

`_POSIX_THREAD_PRIO_PROTECT`
The implementation supports the priority protection option.

Constants for Functions

The following symbolic constant is defined:

`NULL` Null pointer.

The following symbolic constants are defined for the `access()` function:

`R_OK` Test for read permission.

`W_OK` Test for write permission.

`X_OK` Test for execute (search) permission.

`F_OK` Test for existence of file.

The constants `F_OK`, `R_OK`, `W_OK` and `X_OK` and the expressions `R_OK | W_OK, R_OK | X_OK` and `R_OK | W_OK | X_OK` all have distinct values.

The following symbolic constants are defined for the `confstr()` function:

`_CS_PATH`
If the ISO POSIX-2 standard is supported, this is the value for the `PATH` environment variable that finds all standard utilities. Otherwise the meaning of this value is unspecified.

`_CS_XBS5_ILP32_OFF32_CFLAGS`
If `sysconf(_SC_XBS5_ILP32_OFF32)` returns –1, the meaning of this value is unspecified. Otherwise, this value is the set of initial options to be given to the `cc` and `c89` utilities to build an application using a programming model with 32-bit `int`, `long`, `pointer`, and `off_t` types.

`_CS_XBS5_ILP32_OFF32_LDFLAGS`
If `sysconf(_SC_XBS5_ILP32_OFF32)` returns –1, the meaning of this value is unspecified. Otherwise, this value is the set of final options to be given to the `cc` and `c89` utilities to build an application using a programming model with 32-bit `int`, `long`, `pointer`, and `off_t` types.

`_CS_XBS5_ILP32_OFF32_LIBS`
If `sysconf(_SC_XBS5_ILP32_OFF32)` returns –1, the meaning of this value is unspecified. Otherwise, this value is the set of libraries to be given to the `cc` and `c89` utilities to build an application using a programming model with 32-bit `int`, `long`, `pointer`, and `off_t` types.

`_CS_XBS5_ILP32_OFF32_LINTFLAGS`
If `sysconf(_SC_XBS5_ILP32_OFF32)` returns –1, the meaning of this value is unspecified. Otherwise, this value is the set of options to be given to the `lint` utility to check application source using a programming model with 32-bit `int`, `long`, `pointer`, and `off_t` types.

`_CS_XBS5_ILP32_OFFBIG_CFLAGS`
If `sysconf(_SC_XBS5_ILP32_OFFBIG)` returns –1, the meaning of this value is unspecified. Otherwise, this value is the set of initial options to be given to the `cc` and `c89` utilities to build an application using a programming model with 32-bit `int`, `long`, and `pointer` types, and an `off_t` type using at least 64 bits.

`_CS_XBS5_ILP32_OFFBIG_LDFLAGS`
If `sysconf(_SC_XBS5_ILP32_OFFBIG)` returns –1, the meaning of this value is unspecified. Otherwise, this value is the set of final options to be given to the `cc` and `c89` utilities to build an application using a programming model with 32-bit `int`, `long`, and `pointer` types, and an `off_t` type using at least 64 bits.

`_CS_XBS5_ILP32_OFFBIG_LIBS`
If `sysconf(_SC_XBS5_ILP32_OFFBIG)` returns –1, the meaning of this value is unspecified. Otherwise, this value is the set of libraries to be given to the `cc` and `c89` utilities to build an application using a programming model with 32-bit `int`, `long`, and `pointer` types, and an `off_t` type using at least 64 bits.

`_CS_XBS5_ILP32_OFFBIG_LINTFLAGS`
If `sysconf(_SC_XBS5_ILP32_OFFBIG)` returns –1, the meaning of this value is unspecified. Otherwise, this value is the set of options to be given to the `lint` utility to check an application using a programming model with 32-bit `int`, `long`, and `pointer` types, and an `off_t` type using at least 64 bits.

`_CS_XBS5_LP64_OFF64_CFLAGS`
If `sysconf(_SC_XBS5_LP64_OFF64)` returns –1, the meaning of this value is unspecified. Otherwise, this value is the set of initial options to be given to the `cc` and `c89` utilities to build an application using a programming model with 64-bit `int`, `long`, `pointer`, and `off_t` types.

`_CS_XBS5_LP64_OFF64_LDFLAGS`
If `sysconf(_SC_XBS5_LP64_OFF64)` returns –1, the meaning of this value is unspecified. Otherwise, this value is the set of final options to be given to the `cc` and `c89` utilities to build an application using a

programming model with 64-bit `int`, `long`, `pointer`, and `off_t` types.

`_CS_XBS5_LP64_OFF64_LIBS`
: If `sysconf(_SC_XBS5_LP64_OFF64)` returns –1, the meaning of this value is unspecified. Otherwise, this value is the set of libraries to be given to the `cc` and `c89` utilities to build an application using a programming model with 64-bit `int`, `long`, `pointer`, and `off_t` types.

`_CS_XBS5_LP64_OFF64_LINTFLAGS`
: If `sysconf(_SC_XBS5_LP64_OFF64)` returns –1, the meaning of this value is unspecified. Otherwise, this value is the set of options to be given to the `lint` utility to check application source using a programming model with 64-bit `int`, `long`, `pointer`, and `off_t` types.

`_CS_XBS5_LPBIG_OFFBIG_CFLAGS`
: If `sysconf(_SC_XBS5_LPBIG_OFFBIG)` returns –1, the meaning of this value is unspecified. Otherwise, this value is the set of initial options to be given to the `cc` and `c89` utilities to build an application using a programming model with an `int` type using at least 32 bits and `long`, `pointer`, and `off_t` types using at least 64 bits.

`_CS_XBS5_LPBIG_OFFBIG_LDFLAGS`
: If `sysconf(_SC_XBS5_LPBIG_OFFBIG)` returns –1, the meaning of this value is unspecified. Otherwise, this value is the set of final options to be given to the `cc` and `c89` utilities to build an application using a programming model with an int type using at least 32 bits and long, pointer, and `off_t` types using at least 64 bits. `_CS_XBS5_LPBIG_OFFBIG_LIBS If sysconf(_SC_XBS5_LPBIG_OFFBIG)` returns –1, the meaning of this value is unspecified. Otherwise, this value is the set of libraries to be given to the `cc` and `c89` utilities to build an application using a programming model with an `int` type using at least 32 bits and `long`, `pointer`, and `off_t` types using at least 64 bits.

`_CS_XBS5_LPBIG_OFFBIG_LINTFLAGS`
: If `sysconf(_SC_XBS5_LPBIG_OFFBIG)` returns –1, the meaning of this value is unspecified. Otherwise, this value is the set of options to be given to the `lint` utility to check application source using a programming model with an `int` type using at least 32 bits and `long`, `pointer`, and `off_t` types using at least 64 bits.

The following symbolic constants are defined for the `lseek()` and `fcntl()` functions (they have distinct values):

`SEEK_SET`
: Set file offset to *offset*.

`SEEK_CUR`
: Set file offset to current plus *offset*.

`SEEK_END`
: Set file offset to EOF plus *offset*.

The following symbolic constants are defined for `sysconf()`:

`_SC_2_C_BIND`
`_SC_2_C_DEV`
`_SC_2_C_VERSION`
`_SC_2_FORT_DEV`
`_SC_2_FORT_RUN`
`_SC_2_LOCALEDEF`
`_SC_2_SW_DEV`
`_SC_2_UPE`
`_SC_2_VERSION`
`_SC_ARG_MAX`
`_SC_ATEXIT_MAX`
`_SC_BC_BASE_MAX`
`_SC_BC_DIM_MAX`
`_SC_BC_SCALE_MAX`
`_SC_BC_STRING_MAX`
`_SC_CHILD_MAX`

```
_SC_CLK_TCK
_SC_COLL_WEIGHTS_MAX
_SC_EXPR_NEST_MAX
_SC_FSYNC
_SC_GETGR_R_SIZE_MAX
_SC_GETPW_R_SIZE_MAX
_SC_IOV_MAX
_SC_JOB_CONTROL
_SC_LINE_MAX
_SC_LOGIN_NAME_MAX
_SC_MAPPED_FILES
_SC_MEMORY_PROTECTION
_SC_NGROUPS_MAX
_SC_OPEN_MAX
_SC_PAGESIZE
_SC_PAGE_SIZE
_SC_PASS_MAX
_SC_RE_DUP_MAX
_SC_SAVED_IDS
_SC_STREAM_MAX
_SC_THREADS
_SC_THREAD_ATTR_STACKADDR
_SC_THREAD_ATTR_STACKSIZE
_SC_THREAD_DESTRUCTOR_ITERATIONS
_SC_THREAD_KEYS_MAX
_SC_THREAD_PRIORITY_SCHEDULING
_SC_THREAD_PRIO_INHERIT
_SC_THREAD_PRIO_PROTECT
_SC_THREAD_PROCESS_SHARED
_SC_THREAD_SAFE_FUNCTIONS
_SC_THREAD_STACK_MIN
_SC_THREAD_THREADS_MAX
_SC_TTY_NAME_MAX
_SC_TZNAME_MAX
_SC_VERSION
_SC_XOPEN_VERSION
_SC_XOPEN_ENH_I18N
_SC_XOPEN_SHM
_SC_XOPEN_UNIX
_SC_XOPEN_XCU_VERSION
_SC_XBS5_ILP32_OFF32
_SC_XBS5_ILP32_OFFBIG
_SC_XBS5_LP64_OFF64
_SC_XBS5_LPBIG_OFFBIG
```

The two constants `_SC_PAGESIZE` and `_SC_PAGE_SIZE` may be defined to have the same value.

The following symbolic constants are defined as possible values for the *function* argument to the `lockf()` function:

`F_LOCK`
: Lock a section for exclusive use.

`F_ULOCK`
: Unlock locked sections.

`F_TEST`
: Test section for locks by other processes.

`F_TLOCK`
: Test and lock a section for exclusive use.

The following symbolic constants are defined for `pathconf()`:

_PC_CHOWN_RESTRICTED
_PC_FILESIZEBITS
_PC_LINK_MAX
_PC_MAX_CANON
_PC_MAX_INPUT
_PC_NAME_MAX
_PC_NO_TRUNC
_PC_PATH_MAX
_PC_PIPE_BUF
_PC_VDISABLE

The following symbolic constants are defined for file streams:

STDIN_FILENO
: File number of stdin. It is 0.

STDOUT_FILENO
: File number of stdout. It is 1.

STDERR_FILENO
: File number of stderr. It is 2.

The following pathnames are defined:

GF_PATH
: Pathname of the group file.

PF_PATH
: Pathname of the passwd file.

The following is declared as a function and may also be defined as a macro [see gethostname(3N-xs)]:

int gethostname(char *address, int address_len);

The following values for constants are defined for this release of System V:

_POSIX_VERSION 199009L
_XOPEN_VERSION 4

Type Definitions

The size_t, ssize_t, uid_t, gid_t, off_t and pid_t types are defined as described in <sys/types.h> [see types(5)].

The useconds_t type is defined as described in <sys/types.h> [see types(5)].

The intptr_t type is defined as described in inttypes.h [see inttypes(5)].

Declarations

The following are declared as functions and may also be defined as macros:

int access(const char *path, int amode);
unsigned int alarm(unsigned int seconds);
int brk(void *addr);
int chdir(const char *path);
int chown(const char *path, uid_t owner, gid_t group);
int chroot(const char *path);
int close(int fildes);
size_t confstr (int name, char *buf, size_t len);
char *crypt(const char *key, const char *salt);
char *ctermid(char *s);
char *cuserid(char *s);
int dup(int fildes);
int dup2(int fildes, int fildes2);
void encrypt(char block[64], int edflag);
int execl(const char *path, const char *arg0, ...);

```
int         execle(const char *file, const char *arg0, ...);
int         execlp(const char *file, const char *arg0, ...);
int         execv(const char *path, char *const argv[]);
int         execve(const char *path, char *const argv[],
                char *const envp[]);
int         execvp(const char *file, char *const argv[]);
void         _exit(int status);
int         fchown(int fildes, uid_t owner, gid_t group);
int         fchdir(int fildes);
pid_t        fork(void);
long int     fpathconf(int fildes, int name);
int         fsync(int fildes);
int         ftruncate(int fildes, off_t length);
int         ftruncate64(int fildes, off64_t length);
char              *getcwd(char *buf, size_t size);
int         getdtablesize(void);
gid_t        getegid(void);
uid_t        geteuid(void);
gid_t        getgid(void);
int         getgroups(int gidsetsize, gid_t grouplist[]);
long         gethostid(void);
char        *getlogin(void);
int         getlogin_r(char *name, size_t namesize);
int               getopt(int argc, char * const argv[],
              const char *optstring);
int         getpagesize(void);
char        *getpass(const char *prompt);
pid_t        getpgid(pid_t pid);
pid_t        getpgrp(void);
pid_t        getpid(void);
pid_t        getppid(void);
pid_t        getsid(pid_t pid);
uid_t        getuid(void);
char        *getwd(char *path_name);
int         isatty(int fildes);
int         lchown(const char *path, uid_t owner, gid_t group);
int         link(const char *path1, const char *path2);
int         lockf(int fildes,int function,off_t size);
int         lockf64(int fildes, int function, off64_t size);
off_t             lseek(int fildes, off_t offset, int whence);
off64_t      lseek64(int fildes, off64_t offset, int whence);
int               nice(int incr);
long int     pathconf(const char *path, int name);
int         pause(void);
int         pipe(int fildes[2]);
ssize_t      pread(int fildes, void *buf, size_t nbyte, off_t offset);
int         pthread_atfork(void (*prepare)(void), void (*prepare)(void),
              void(*child)(void));
ssize_t      pwrite(int fildes, const void *buf, size_t nbyte, off_t offset);
ssize_t      read(int fildes, void *buf, size_t nbyte);
int         readlink(const char *path, char *buf, size_t bufsiz);
int         rmdir(const char *path);
void        *sbrk(intptr_t incr);
```

int setgid(gid_t *gid*);
int setpgid(pid_t *pid*, pid_t *pgid*);
pid_t setpgrp(void);
int setregid(gid_t *rgid*, gid_t *egid*);
int setreuid(uid_t *ruid*, uid_t *euid*);
pid_t setsid(void);
int setuid(uid_t *uid*);
unsigned int sleep(unsigned int *seconds*);
void swab(const void *src, void *dest, ssize_t *nbytes*);
int symlink(const char *path1, const char *path2);
void sync(void);
long int sysconf(int *name*);
pid_t tcgetpgrp(int *fildes*);
int tcsetpgrp(int *fildes*, pid_t *pgrp_id*);
int truncate(const char *path, off_t *length*);
int truncate64(const char *path, off64_t *length*);
char *ttyname(int *fildes*);
int ttyname_r(int *fildes*, char *name, size_t *namesize*);
useconds_t ualarm(useconds_t *useconds*, useconds_t *interval*);
int unlink(const char *path);
int usleep(useconds_t *useconds*);
pid_t vfork(void);
ssize_t write(int *fildes*, const void *buf, size_t *nbyte*);

The following external variables are declared:

```
extern char *optarg;
extern int   optind, opterr, optopt;
```

Version Test Macros:

_LFS_LARGEFILE
: Is defined to be 1 if the implementation supports the interfaces as specified in "2.2.1 Changes to System Interfaces" [LFS specification, see lfs(5)] except that implementations need not provide the asynchronous I/O interfaces: aio_read(), aio_write(), and lio_listio().

_LFS_ASYNCHRONOUS_IO
: Is defined to be 1 if the implementation supports the asynchronous IO interfaces: aio_read(), aio_write(), and lio_listio() as specified in "2.2.1 Changes to System Interfaces" [LFS specification, see lfs(5)].

_LFS64_ASYNCHRONOUS_IO
: Is defined to be 1 if the implementation supports all the transitional extensions listed in "3.1.1.1.1 Asynchronous I/O Interfaces" and "3.1.2.2 <aio.h>" [LFS specification, see lfs(5)].

_LFS64_LARGEFILE
: Is defined to be 1 if the implementation supports all the transitional extensions listed in "3.1.1.1.3 Other Interfaces", "3.1.1.2 fcntl()", "3.1.1.3 open()" and "3.1.2 Transitional Extensions to Headers", except changes specified in "3.1.2.2 <aio.h>" and "3.1.2.6 <stdio.h>" [LFS specification, see lfs(5)] need not be supported.

_LFS64_STDIO
: Is defined to be 1 if the implementation supports all the transitional extensions listed in "3.1.1.1.2 STDIO Interfaces" and "3.1.2.6 <stdio.h>" [LFS specification, see lfs(5)].

  If _LFS64_STDIO is not defined to be 1 and the underlying file description associated with stream has O_LARGEFILE set then the behavior of the standard I/O functions is unspecified.

Constants for functions:

_CS_LFS_CFLAGS
: For confstr().

_CS_LFS_LDFLAGS
For confstr().

_CS_LFS_LIBS
For confstr().

_CS_LFS_LINTFLAGS
For confstr().

_CS_LFS64_CFLAGS
For confstr().

_CS_LFS64_LDFLAGS
For confstr().

_CS_LFS64_LIBS
For confstr().

_CS_LFS64_LINTFLAGS
For confstr().

SEE ALSO

access(2), alarm(2), brk(2), chdir(2), chown(2), chroot(2), close(2), dup(2), exec(2), exit(2), fcntl(2), fork(2), fpathconf(2), fsync(2), getgroups(2), getpid(2), getsid(2), getuid(2), link(2), lseek(2), nice(2), pause(2), pipe(2), read(2), readlink(2), rmdir(2), setpgid(2), setpgrp(2), setsid(2), setuid(2), symlink(2), sync(2), unlink(2), vfork(2), write(2), getdtablesize(3), getpagesize(3), getwd(3), ualarm(3), usleep(3), confstr(3C), crypt(3C), getcwd(3C), getlogin(3C), getpass(3C), lockf(3C), setregid(3C), setreuid(3C), sleep(3C), swab(3C), sysconf(3C), tcsetpgrp(3C), truncate(3C), ttyname(3C), gethostid(3N), gethostname(3N-xs), ctermid(3S), cuserid(3S), group(4), limits(4), passwd(4), inttypes(5), lfs(5), termios(5), types(5).

vfstab(4)

NAME

`vfstab` - table of file system defaults

SYNOPSIS

```
#include <sys/fstyp.h>
#include <sys/param.h>
#include <sys/vfstab.h>
```

DESCRIPTION

The file `etc/vfstab` contains the defaults for each file system. The information is in the following structure, defined in `<sys/vfstab.h>`:

```
struct vfstab
{
    char *vfs_special;
    char *vfs_fsckdev;
    char *vfs_mountp;
    char *vfs_fstype;
    char *vfs_fsckpass;
    char *vfs_automnt;
    char *vfs_mntopts;
};
```

The fields in the table are separated by blanks. Comment lines begin with a "#". Blank lines are ignored. A minus sign "–" in a field means that no action takes place or no entry has been made. There are 7 fields or columns in total:

Column 1: "special"
: Block-oriented or resource name

Column 2: "fsckdev"
: Raw device for `fsck`(1M)

Column 3: "mountp"
: Default mount directory

Column 4: "fstype"
: Name of the file system type

Column 5: "fsckpass"
: Entry specifying which file systems `fsck`(1M) should check automatically: `0-9` (`0` = default file systems, `1-9` other file systems), –□(=□none)

Column 6: "automnt"
: Entry specifying whether the file system should be mounted automatically: `yes`, `no` or – (the `root` file system "/" is always mounted)

Column 7: "mntopts"
: Entry specifying which specific `mount` options (`-o`) the file system is to be automatically mounted with (if any).

The `getvfsent` family of routines are used to read and write to `/etc/vfstab`.

SEE ALSO

`fsck`(1M), `mount`(1M), `setmnt`(1M), `getvfsent`(3C).

Miscellaneous Facilities(5)

Introduction

System Administrator's Reference Manual

intro_sarm5(5)

NAME

`intro_sarm5` - introduction to miscellany

DESCRIPTION

This section describes miscellaneous facilities such as macro packages, conversion tables, tables of Shell special characters and regular expressions, etc.

eqnchar(5) - XXsetint(5)

System Administrator's Reference Manual

eqnchar(5)

NAME

`eqnchar` - special character definitions for eqn and neqn

SYNOPSIS

`eqn /usr/pub/eqnchar` [*option* ...] [—] [*file* ...] | `troff` [*option* ...]

`neqn /usr/pub/eqnchar` [*option* ...] [—] [*file* ...] | `nroff` [*option* ...]

DESCRIPTION

`/usr/pub/eqnchar` contains the following `troff(1M)` and `nroff(1M)` character definitions that are not ordinarily available on a phototypesetter or printer. These definitions are primarily intended for use with `eqn(1M)` and `neqn(1M)` .

`eqnchar` contains definitions for the following characters:

ciplus	⊕	‖	‖	square	□
citimes	⊗	langle	⟨	circle	○
wig	~	rangle	⟩	blot	
-wig	≈	hbar	ℏ	bullet	•
>wig	≳	ppd	∟	prop	∝
<wig	≲	<->	→	empty	∅
=wig	≑	<=>	⇔	member	∈
star	∗	|<	∤	nomem	∉
bigstar	✶	|>	∤	cup	∪
=dot	≐	ang	∠	cap	∩
orsign	∨	rang	∟	incl	⊑
andsign	∧	3dot		subset	⊂
=del	≜	thf	∴	supset	⊃
oppA	⊥	quarter	¼	!subset	⊈
oppE	∄	3quarter	¾	!supset	⊉
angstrom	Å	degree	°	scrL	ℓ
==<	≤	==>	≥		

`apseqnchar` is a version of `eqnchar` tailored for the particular output device Autologic APS-5; output will not look optimal on other devices.

`cateqnchar` is more "device-independent", and should produce output that looks reasonable on any device supported by `troff(1M)` .

By default, `eqnchar` is linked to `cateqnchar`.

FILES

`/usr/pub/eqnchar`
`/usr/pub/apseqnchar`
`/usr/pub/cateqnchar`

SEE ALSO

`eqn`(1M), `neqn`(1M), `nroff`(1M), `troff`(1M), `roff-charset`(5).

REFERENCES

C. Scrocca, *New Graphic Symbols for NEQN and EQN*

B. W. Kernighan, L. L. Cherry, *Typesetting Mathematics — User's Guide* (2nd Ed.)

System Administrator's Reference Manual

expressions(5)

NAME

`expressions` - regular expressions

DESCRIPTION

Regular expressions are used for scanning a text for strings which match a defined pattern. A regular expression stands for a set of characters or character strings. Each character string in this set is said to be matched by the regular expression. A pattern is constructed from one or more regular expressions.

A regular expression comprises a string of characters, which can be further classified into:

- ordinary characters

  All characters in the character set, except for the newline character and metacharacters, are ordinary characters. Within a pattern, ordinary characters match themselves, i.e. the pattern `abc` will match only those strings that contain the character sequence `abc` anywhere in them.

- metacharacters

  Metacharacters do not match themselves, but have a special meaning, which is explained below. Metacharacters preceded by a backslash `\` lose their special meaning.

There are two forms of regular expression:

- simple regular expressions
- extended regular expressions

The syntax of these forms of regular expression is described in the following sections.

SIMPLE REGULAR EXPRESSIONS

The pattern to be searched for in a text can be made up of any single expressions. The following single expressions can be used if a command supports simple regular expressions.

Single characters, collating elements

Expression	Meaning
c	The character *c*, where *c* must not be a special character (metacharacter). □ n*Example*: `a` matches `a`
c	The character *c*, where *c* can be any character other than ( ) { } `1 2 3 4 5 6 7 8 9`. □ A regular expression in the form *c* is meaningful if *c* is a metacharacter. *c* then stands for the character *c* itself. □ *Example*: `\a` matches `a`, `\*` matches `*`
[.*cc*.]	(Collating symbol; only within [ ]) Multi-character collating elements must be represented in this form to distinguish them from ordinary characters. An expression of this type is collated as a single character. *cc* has to be defined as a valid collating element in the internationalized environment. □ *Example*: □ In the Spanish locale `LANG=LC_COLLATE=Es_SP.88591` `ch` is a valid collating element: in Spanish, `ch` is treated as a single letter collating between `c` and `d`. This letter must be represented in the form [`.ch.`] to distinguish it from the two-letter string `ch`.

Groups of characters, classes

Expression	Meaning

.	Any character. □ *Example*: . matches a, x, *, ...
[*s*]	Any character from the character string *s*. *s* can also be a character class. □ *Example*: [mz] matches m, z □ *Warning*: □ Metacharacters that have a special meaning in bracketed expressions (], –, ^) are treated as normal characters if they are placed at a particular position in the bracketed expression, i.e. □] .i1 +3n at the first position .i2 -3n – .i1 +3n at the first or last position .i2 -3n ^ .i1 +3n at any position except the first or last. .i2 -3n
[*c1*–*c2*]	Any character in the range between *c1* and *c2* in accordance with the currently valid collating sequence (*c1* and *c2* inclusive). *c1* and *c2* can also be expressions for equivalence classes [=*c*=] or collating symbols [. *cc* .]. □ *Example*: □ In the German locale LANG=LC_COLLATE=De_DE.88591 [a–d] matches the characters a, ä, b, c, d, while in the Spanish locale LANG=LC_COLLATE=Es_SP.88591, it matches the characters a, b, c, ch, d.
[*s1c1–c2s2*]	The two forms can be combined.
[^*s*]	Any character not contained in the character string *s*. □ *Example*: [^xyz] matches every character excluding x, y, z.
[^*c1–c2*]	Any character not in the range between *c1* and *c2*. □ *Example*: □ [^0–9] matches every character excluding 0, 9, and the characters in the collating sequence between 0 and 9.
[^*s1c1–c2s2*]	The two forms can be combined.

[:*class*:]	(Character class expression; only within [ ]) Any character from the character class *class*. *class* can be: □ alpha .i1 +5m any letter .i2 -5m upper .i1 +5m any uppercase letter .i2 -5m lower .i1 +5m any lowercase letter .i2 -5m digit .i1 +5m any decimal digit (0 through 9) .i2 -5m xdigit .i1 +5m any hexadecimal digit (0 through 9, a through f, and A through F) .i2 -5m alnum .i1 +5m any alphanumeric character (letters and digits) .i2 -5m space .i1 +5m any character producing white space in displayed text (e.g. blanks or tabs) .i2 -5m blank .i1 +5m blanks or tabs .i2 -5m punct .i1 +5m any separator .i2 -5m print .i1 +5m any printable character (including the characters in space) .i2 -5m graph .i1 +5m any printable character with a visible representation (excluding the characters in space) .i2 -5m cntrl .i1 +5m any control character .i2 -5m □ *Example*: □ In the German locale LANG=LC_CTYPE=De_DE.88591 the characters ä, ö, ü, ß, a, ..., z match the regular expression [[:lower:]]. The character è does not belong to lower or alpha, but is a printable character.
[=*c*=]	(Equivalence class expression; only within [ ]) Any character or collating element defined as having the same relative order as *c*. *c* must not be an equals sign = or a right square bracket]. □ *Example*: □ In the German locale LANG=LC_COLLATE=De_DE.88591 the characters u and ü form an equivalence class. Consequently the characters u and ü match the regular expression [[=u=]]. In this locale, the regular expressions [[=u=]v], [[=ü=]v], and [uüv] are synonyms.

Concatenation

Single expressions can be concatenated. All concatenated expressions together describe the pattern to be searched for in a text.

Expression	Meaning
rx	An occurrence of a character string matching the regular expression *r*, followed by a character string matching the regular expression *x*. □ *Example*: `[ab].` matches `ax`, `a3`, `a*`, `bz`, ...

Repeats

Expressions that describe single characters or groups of characters can be repeated, as can references back to subexpressions.

Expression	Meaning
r`*`	Zero, one, or more occurrences of the regular expression *r*. □ *Example*: `a*` matches nothing, `a`, `aa`, `aaa`, ...
r`\{`*m*`,`*n*`\}`	At least *m* and at most *n* occurrences of the regular expression *r*. □ *Example*: `a\{1,2\}` matches `a` or `aa`
r`\{`*m*`\}`	Exactly *m* occurrences of the regular expression *r*. □ *Example*: `a\{3\}` matches `aaa`
r`\{`*m*`,\}`	At least *m* occurrences of the regular expression *r*. □ *Example*: `a\{3,\}` matches `aaa`, `aaaa`, `aaaaa`, ...

Anchoring

Patterns can be "anchored" at the start or the end of a line.

Expression	Meaning
`^`*r*	A character string appearing at the start of a line, that matches the regular expression *r*, i.e. straight after a newline character or at the start of the file. □ *Example*: `^[aA]pple` matches `apple` or `Apple` at the start of a line.
r`$`	A character string appearing at the end of a line, that matches the regular expression *r*, i.e. directly before a newline character. □ *Example*: `[bB]arge$` matches `barge` or `Barge` at the end of a line.

Subexpressions and references

Parts of a pattern can be combined as a subexpression. This subexpression can then be repeated at a later position in the pattern by means of a reference. The reference always stands for the same character string as the subexpression.

Expression	Meaning
`\(`*x*`\)`	The regular expression *x* is identified as a subexpression. It matches all character strings that match the regular expression *x*. □ *Example*: `\(aa*\)` matches `a`, `aa`, `aaa`, ...
`\`*n*	*n* is an integer between 1 and 9. `\`*n* is reference to the *n*th subexpression *x* in a pattern. *x* must be placed before the reference in the pattern. `\`*n* matches the same character string as *x*. □ *Example*: □ `\(aa*\)_\1` matches `a_a`, `aa_aa`, `aaa_aaa`, ...; `\(a\(b\)\)\2` matches `abb`

Grouping, alternatives

Only exist for extended regular expressions.

EXTENDED REGULAR EXPRESSIONS

The pattern to be searched for in a text can be made up of any single expressions. The following single expressions can be used if a command supports extended regular expressions.

Single characters, collating elements

As for simple regular expressions.

Groups of characters, classes

As for simple regular expressions.

Concatenation

As for simple regular expressions.

Repeats

Expressions that describe single characters or groups of characters can be repeated, as can groupings and alternatives.

Expression	Meaning
r`*`	Zero, one, or more occurrences of the regular expression *r*. □ *Example*: `a*` matches nothing, `a`, `aa`, `aaa`, ...
r`+`	One or more occurrences of the regular expression *r*. □ *Example*: `u+` matches `u`, `uu`, `uuu`, ...
r`?`	Zero or one occurrences of the regular expression *r*. □ *Example*: `u?` matches nothing or `u`
r`{`*m*`,`*n*`}`	At least *m* and at most *n* occurrences of the regular expression *r*. □ *Example*: `a{1,2}` matches `a` or `aa`
r`{`*m*`}`	Exactly one occurrence *m* of the regular expression *r*. □ *Example*: `a{3}` matches `aaa`
r`{`*m*`,}`	At least *m* occurrences of the regular expression *r*. □ *Example*: `a{3,}` matches `aaa`, `aaaa`, `aaaaa`, ...

Anchoring

As for simple regular expressions.

Subexpressions and references

Only exist for simple regular expressions.

Grouping, alternatives

Expression	Meaning
`(`*rx*`)`	The regular expressions *r* and *x* are combined in a group that matches all character strings matching the regular expression *rx*. □ *Example*: □ `(ok(abc))` matches `okabc`; `(au)*` matches nothing or `au`, `auau`, ...
`(`*r1*`|`*r2*`)`	Character strings that match the regular expression *r1* or *r2*. □ *Example*: `(ok|ko)` matches `ok` or `ko`

PRECEDENCE

The following tables show the precedence of operators in regular expressions. The operators are collated in descending order from highest precedence to lowest precedence.

Precedence for simple regular expressions

Symbols from the internationalized environment	`[= =][: :][. .]`
Quoting characters	`\`*character*
Bracketed expressions	`[ ]`
Subexpressions, references	`\( \) \`*n*
Repeat	`* \{`*m*`,`*n*`\}`
Concatenation	`rx`
Anchoring	`^ $`

Precedence for extended regular expressions

Symbols from the internationalized environment	`[= =][: :][. .]`
Quoting characters	`\`*character*
Bracketed expressions	`[ ]`
Grouping	`( )`
Repeat	`* ? + {`*m*`,`*n*`}`
Concatenation	`rx`
Anchoring	`^ $`
Alternatives	`|`

Commands with regular expressions

The following table is an overview of the commands that process regular expressions.

Command	Type of regular expressions
`apropos`	extended
`awk`	extended internationalized
`bfs`	simple
`csplit`	simple
`e`	simple
`ed`	simple
`egrep`	extended
`ex`	simple
`expr`	simple
`extract`	simple
`findman`	extended
`grep`	simple

lex	extended
man	simple
nl	simple
pg	simple
ed	simple
vi	simple
whatis	extended

LOCALE

In bracketed regular expressions the LC_COLLATE environment variable determines the meaning of metacharacters, equivalence classes, and collating elements, while the LC_CTYPE environment variable determines the meaning of character classes.

If LC_COLLATE or LC_CTYPE is undefined or defined as a null string, the value of LANG is taken as the default value for the unset or empty variable. If LANG is also undefined or defined as a null string, the system behaves as if it has not been internationalized.

If one of the variables has a value that is invalid for the internationalized environment, the system behaves as if none of the variables has been set.

The LC_ALL environment variable determines the entire internationalized environment. LC_ALL takes precedence over all other environment variables that affect internationalization.

SEE ALSO

awk(1), bfs(1), csplit(1), e(1), ed(1), egrep(1), ex(1), expr(1), extract(1), grep(1), lex(1), man(1), nl(1), pg(1), regcmp(1), sed(1), vi(1), regcomp(3C), regex(3C), regcmp(3G), regexpr(3G), regex(5), regexp(5).

System Administrator's Reference Manual

font(5)

NAME

`font` - typesetter description tables

SYNOPSIS

`troff` –T *name* ...

DESCRIPTION

Typesetter tables for the *name* device are located in the `/usr/lib/font/dev`*name* directory. The tables are used by `troff` and most postprocessors.

The tables are ASCII and are read one line at a time. The first word on each line is a keyword. The remainder of the line is the value associated with that keyword. Lines that begin with unrecognized keywords are ignored. Programs that read typesetter tables are free to recognize their own collection of keywords provided they parse the standard set (below) properly.

A general description of the typesetter can be found in a file named `DESC`. The standard set of `DESC` file keywords includes:

`res` Device resolution in dots per inch.

`hor` Smallest unit of horizontal motion.

`vert` Smallest unit of vertical motion.

`sizes`
: Available point sizes. List must be given in increasing numerical order, be terminated with a 0, and can span more than one line.

`unitwidth`
: Widths in font tables are given at this point size.

`fonts`
: Number of font positions followed by the list of default fonts. The list can span more than one line.

`charset`
: List of special characters available in the typesetter. Starts on the next line and continues to the end of the `DESC` file.

The remaining files describe the fonts available on device *name*. The standard set of font table keywords is:

`name` Font name as known to `troff`.

`special`
: Indicates the font is special.

`ligatures`
: Which ligatures (if any) exist of this font. List is terminated by 0.

`spacewidth`
: Width of a space on this font.

`charset`
: Character data for this font. Starts on the next line and continues to the end of the font table. Character data includes name, width, ascender/descender, and code fields. A double quote in the width field indicates a synonym for the preceding character.

SEE ALSO

`dpost`(1M), `troff`(1M), `troff`(5).

REFERENCE

B. W. Kernighan, *A Typesetter-Independent Troff*

System Administrator's Reference Manual

iconv_tab(5)

NAME

`iconv_tab -` code set conversion tables

DESCRIPTION

The following code set conversions are supported:

Code	Symbol	Target Code	Symbol	comment
ISO 646	646	ISO 8859–1	8859	US ASCII
ISO 646de	646de	ISO 8859–1	8859	German
ISO 646da	646da	ISO 8859–1	8859	Danish
ISO 646en	646en	ISO 8859–1	8859	English ASCII
ISO 646es	646es	ISO 8859–1	8859	Spanish
ISO 646fr	646fr	ISO 8859–1	8859	French
ISO 646it	646it	ISO 8859–1	8859	Italian
ISO 646sv	646sv	ISO 8859–1	8859	Swedish
ISO 8859–1	8859	ISO 646	646	7 bit ASCII
ISO 8859–1	8859	ISO 646de	646de	German
ISO 8859–1	8859	ISO 646da	646da	Danish
ISO 8859–1	8859	ISO 646en	646en	English ASCII
ISO 8859–1	8859	ISO 646es	646es	Spanish
ISO 8859–1	8859	ISO 646fr	646fr	French
ISO 8859–1	8859	ISO 646it	646it	Italian
ISO 8859–1	8859	ISO 646sv	646sv	Swedish

The conversions are performed according to the tables following. All values in the tables are given in octal.

ISO 646 (US ASCII) to ISO 8859-1

For the conversion of ISO 646 to ISO 8859-1 all characters in ISO 646 can be mapped unchanged to ISO 8859-1

ISO 646de (GERMAN) to ISO 8859-1

For the conversion of ISO 646de to ISO 8859-1 all characters not in the following table are mapped unchanged.

Conversions Performed

ISO 646de	ISO 8859–1
100	247
133	304
134	326
135	334

173	344
174	366
175	374
176	337

ISO 646da (DANISH) to ISO 8859-1

For the conversion of ISO 646da to ISO 8859-1 all characters not in the following table are mapped unchanged.

Conversions Performed

ISO 646da	ISO 8859–1
133	306
134	330
135	305
173	346
174	370
175	345

ISO 646en (ENGLISH ASCII) to ISO 8859-1

For the conversion of ISO 646en to ISO 8859-1 all characters not in the following table are mapped unchanged.

Conversions Performed

ISO 646en	ISO 8859–1
043	243

ISO 646fr (FRENCH) to ISO 8859-1

For the conversion of ISO 646fr to ISO 8859-1 all characters not in the following table are mapped unchanged.

Conversions Performed

ISO 646fr	ISO 8859–1
043	243
100	340
133	260
134	347
135	247
173	351
174	371
175	350
176	250

ISO 646it (ITALIAN) to ISO 8859-1

For the conversion of ISO 646it to ISO 8859-1 all characters not in the following table are mapped unchanged.

Conversions Performed

ISO 646it	ISO 8859–1
043	243
100	247
133	260
134	347
135	351
140	371
173	340
174	362
175	350
176	354

ISO 646es (SPANISH) to ISO 8859-1

For the conversion of ISO 646es to ISO 8859-1 all characters not in the following table are mapped unchanged.

Conversions Performed

ISO 646es	ISO 8859–1
100	247
133	241
134	321
135	277
173	260
174	361
175	347

ISO 646sv (SWEDISH) to ISO 8859-1

For the conversion of ISO 646sv to ISO 8859-1 all characters not in the following table are mapped unchanged.

Conversions Performed

ISO 646sv	ISO 8859–1
100	311
133	304
134	326
135	305
136	334

140	351
173	344
174	366
175	345
176	374

ISO 8859-1 to ISO 646 (ASCII)

For the conversion of ISO 8859-1 to ISO 646 all characters not in the following table are mapped unchanged.

Conversions to Underscore '_'(137)

200	201	202	203	204	205	206	207
210	211	212	213	214	215	216	217
220	221	222	223	224	225	226	227
230	231	232	233	234	235	236	237
240	241	242	243	244	245	246	247
250	251	252	253	254	255	256	257
260	261	262	263	264	265	266	267
270	271	272	273	274	275	276	277
300	301	302	303	304	305	306	307
310	311	312	313	314	315	316	317
320	321	322	323	324	325	326	327
330	331	332	333	334	335	336	337
340	341	342	343	344	345	346	347
350	351	352	353	354	355	356	357
360	361	362	363	364	365	366	367
370	371	372	373	374	375	376	377

ISO 8859-1 to ISO 646de (GERMAN)

For the conversion of ISO 8859-1 to ISO 646de all characters not in the following tables are mapped unchanged.

Conversions Performed

ISO 8859−1	ISO 646de
247	100
304	133
326	134
334	135
337	176
344	173
366	174

374	175

Converted to Underscore '_'(137)

100	133	134	135	173	174	175	176
200	201	202	203	204	205	206	207
210	211	212	213	214	215	216	217
220	221	222	223	224	225	226	227
230	231	232	233	234	235	236	237
240	241	242	243	244	245	246	
250	251	252	253	254	255	256	257
260	261	262	263	264	265	266	267
270	271	272	273	274	275	276	277
300	301	302	303		305	306	307
310	311	312	313	314	315	316	317
320	321	322	323	324	325		327
330	331	332	333		335	336	337
340	341	342	343		345	346	347
350	351	352	353	354	355	356	357
360	361	362	363	364	365		367
370	371	372	373		375	376	377

ISO 8859-1 to ISO 646da (DANISH)

For the conversion of ISO 8859-1 to ISO 646da all characters not in the following tables are mapped unchanged.

Conversions Performed

ISO 8859–1	ISO 646da
305	135
306	133
330	134
345	175
346	173
370	174

Converted to Underscore '_'(137)

	133	134	135	173	174	175	
200	201	202	203	204	205	206	207
210	211	212	213	214	215	216	217
220	221	222	223	224	225	226	227

230	231	232	233	234	235	236	237
240	241	242	243	244	245	246	247
250	251	252	253	254	255	256	257
260	261	262	263	264	265	266	267
270	271	272	273	274	275	276	277
300	301	302	303	304			307
310	311	312	313	314	315	316	317
320	321	322	323	324	325	326	327
	331	332	333	334	335	336	337
340	341	342	343	344			347
350	351	352	353	354	355	356	357
360	361	362	363	364	365	366	367
	371	372	373	374		376	377

ISO 8859-1 to ISO 646en (ENGLISH ASCII)

For the conversion of ISO 8859-1 to ISO 646en all characters not in the following tables are mapped unchanged.

Conversions Performed

ISO 8859–1	ISO 646en
243	043

Converted to Underscore '_'(137)

043							
200	201	202	203	204	205	206	207
210	211	212	213	214	215	216	217
220	221	222	223	224	225	226	227
230	231	232	233	234	235	236	237
240	241	242		244	245	246	247
250	251	252	253	254	255	256	257
260	261	262	263	264	265	266	267
270	271	272	273	274	275	276	277
300	301	302	303	304	305	306	307
310	311	312	313	314	315	316	317
320	321	322	323	324	325	326	327
330	331	332	333	334	335	336	337
340	341	342	343	344	345	346	347
350	351	352	353	354	355	356	357
360	361	362	363	364	365	366	367

370	371	372	373	374	375	376	377

ISO 8859-1 to ISO 646fr (FRENCH)

For the conversion of ISO 8859-1 to ISO 646fr all characters not in the following tables are mapped unchanged.

Conversions Performed

ISO 8859–1	ISO 646fr
243	043
247	135
250	176
260	133
340	100
347	134
350	175
351	173
371	174

Converted to Underscore '_'(137)

043							
100	133	134	135	173	174	175	176
200	201	202	203	204	205	206	207
210	211	212	213	214	215	216	217
220	221	222	223	224	225	226	227
230	231	232	233	234	235	236	237
240	241	242		244		245	246
	251	252	253	254	255	256	257
	261	262	263	264	265	266	267
270	271	272	273	274	275	276	277
300	301	302	303	304	305	306	307
310	311	312	313	314	315	316	317
320	321	322	323	324	325	326	327
330	331	332	333	334	335	336	337
	341	342	343	344		345	346
		352	353	354	355	356	357
360	361	362	363	364	365	366	367
370		372	373	374	375	376	377

ISO 8859-1 to ISO 646it (ITALIAN)

For the conversion of ISO 8859-1 to ISO 646it all characters not in the following tables are mapped

unchanged.

Conversions Performed	
ISO 8859−1	ISO 646it
243	043
247	100
260	133
340	173
347	134
350	175
351	135
354	176
362	174
371	140

Converted to Underscore '_'(137)

043							
100	133	134	135	173	174	175	176
200	201	202	203	204	205	206	207
210	211	212	213	214	215	216	217
220	221	222	223	224	225	226	227
230	231	232	233	234	235	236	237
240	241	242		244		245	246
250	251	252	253	254	255	256	257
	261	262	263	264	265	266	267
270	271	272	273	274	275	276	277
300	301	302	303	304	305	306	307
310	311	312	313	314	315	316	317
320	321	322	323	324	325	326	327
330	331	332	333	334	335	336	337
	341	342	343	344		345	346
		352	353	354	355	356	357
360	361		363	364	365	366	367
370		372	373	374	375	376	377

ISO 8859-1 to ISO 646es (SPANISH)

For the conversion of ISO 8859-1 to ISO 646es all characters not in the following tables are mapped unchanged.

Converted to Underscore '_'(137)

100	133	134	135	173	174	175	
200	201	202	203	204	205	206	207
210	211	212	213	214	215	216	217
220	221	222	223	224	225	226	227
230	231	232	233	234	235	236	237
240		242	243	244		245	246
250	251	252	253	254	255	256	257
	261	262	263	264	265	266	267
270	271	272	273	274		275	276
300	301	302	303	304	305	306	307
310	311	312	313	314	315	316	317
320		322	323	324	325	326	327
330	331	332	333	334	335	336	337
340	341	342	343	344		345	346
350	351	352	353	354	355	356	357
360		362	363	364	365	366	367
370	371	372	373	374	375	376	377

ISO 8859-1 to ISO 646sv (SWEDISH)

For the conversion of ISO 8859-1 to ISO 646sv all characters not in the following tables are mapped unchanged.

Conversions Performed

ISO 8859–1	ISO 646sv
304	133
305	135
311	100
326	134
334	136
344	173
345	175
351	140
366	174
374	176

Converted to Underscore '_'(137)

100	133	134	135	136	140		
	173	174	175	176			
200	201	202	203	204	205	206	207

210	211	212	213	214	215	216	217
220	221	222	223	224	225	226	227
230	231	232	233	234	235	236	237
240	241	242	243	244	245	246	247
250	251	252	253	254	255	256	257
260	261	262	263	264	265	266	267
270	271	272	273	274	275	276	277
300	301	302	303			306	307
310		312	313	314	315	316	317
320	321	322	323	324	325		327
330	331	332	333		335	336	337
340	341	342	343			346	347
350		352	353	354	355	356	357
360	361	362	363	364	365		367
370	371	372	373		375	376	377

FILES

`/usr/lib/iconv/iconv_data`
: lists the conversions supported

`/usr/lib/iconv/*.t`
: conversion tables

SEE ALSO

`iconv`(1).

System Administrator's Reference Manual

log3(5)

NAME

log3 - log system Logging V3.0

DESCRIPTION

The task of the log system Logging V3.0 which can be implemented on all systems with the SINIX operating system (V5.41 and later) is to gather information about processing, status, and error situations in system and application programs (applications) in log files, and to evaluate these using a function. With SINIX V5.43A or Reliant UNIX, Logging V3.0 can react to incoming log messages as soon as programs are started.

A number of applications can transfer their log data to the log server for further processing. A C programming interface and a command log3logger(1M) are available to the applications for transferring data.

"Named pipes", "sockets" or "streams" can function as communications mechanisms. It is irrelevant to the application which option is chosen, since the log server interface is not affected.

The application programmer has extensive freedom with regard to the structure in which the user data is transferred to the log server. In the interests of effective evaluation of the log information there is little point in logging large volumes of data (>1400 characters). Examples of useful log information could be the following, for example:

- which application was started by which user at what time;
- when a particular administration file was edited, and by whom;
- with which status an application was aborted.

For identification purposes, the log data is preceded by a header [see log3header(5)]. User data and headers together make up the log information. The log information (log message) is written to one of the log files by the log server. The system administrator determines to which log file the message should be written, whether the message should also be passed on to Event Reporting DSM-ER, whether the message should start the reaction programs, and so on. This is done by editing the administration file log3admin(4) , to which only the system administrator or a user with similar authorization has access. In particular, the system administrator assigns the log files to the applications.

To facilitate more precise classification of the log information, each application should provide the following information on the interface:

- Error number/severity of errors: This parameter can be freely defined by the application to enable a differentiated analysis of the log information. The error numbers should be unique within the application. The classification of syslog(3C) is offered as an error level.
- User text=module name: The text can be freely defined by the application. A significant text no longer than 16 characters should be selected to enable log messages to be located unambiguously. Depending on the particular case, the package names, driver names, or other a priori unique names should be selected.
- Component number: A classification of all logging applications. You can apply to the system integration for a new component number. The number range is divided up to assign new component numbers. The log messages transferred to Logging V3.0 over syslogd(1M) have special component numbers:

  4294967291 =
  NVRAM

  4294967292 =
  driver

  4294967293 =
  kernel

  4294967294 =
  remaining messages from syslogd(1M)

Precisely one log message should describe at least one combination of error number/module name or error

number/component number. The application can transfer the following as additional parameters:

- Structure ID: The structure ID controls the representation of user data by evaluating log3display(1M) or log3eval(1M). The ID and the related output format are defined by the system administrator in the description file log3struct(4).
- Release version: Provides more detailed specifications of the logging applications.

Logging V3.0 enables the selection and filtering of log information using the header information. The header information therefore should be as complete and unambiguous as possible. The preparation of the log data for log information, saving of this log information, transfer to Event Reporting DSM-ER, and starting of reaction programs when events occur run automatically.

FILES

`LOGPATH=/opt/log3`
: Installation directory for Logging V3.0

`$LOGPATH/log3admin`
: Configuration file: Assignment of applications to log files

`$LOGPATH/log3struct`
: Configuration file: Assignment of structure IDs to output information

`$LOGBOOKS/`*name*`[0-8]log`
: Log files

`$LOGPATH/`*name*`[0-8]log`
: Log files

SEE ALSO

logger(1), logger(1-ucb), errlogdo(1M), errlogget(1M), errlogreplace(1M), log3adm(1M), log3display(1M), log3do(1M), log3eval(1M), log3event_convert(1M), log3get(1M), log3get_pevman(1M), log3logger(1M), log3replace(1M), log3set_pevman(1M), log3str(1M), syslogd(1M), syslog(3C), log3admin(4), log3status(4), log3struct(4), syslog.conf(4), log3close(5), log3header(5), log3memcpy(5), log3open(5), log3printf(5), log3write(5).

System Administrator's Reference Manual

log3close(5)

NAME

`log3close -` shut down communication connection

SYNOPSIS

```
void log3close()
```

DESCRIPTION

The function `log3close` shuts down the communication connection to the log server.

Thus, the pipe to the log server is closed if it was open.

SEE ALSO

`log3`(5), `log3memcpy`(5), `log3open`(5), `log3printf`(5), `log3write`(5).

System Administrator's Reference Manual

log3header(5)

NAME

log3header - conventions for log entries

DESCRIPTION

Messages to be transferred to Logging V3.0 using the syslog(3C) routine or the logger(1-ucb) command must be assigned headers that can be evaluated by Logging so that they can be identified.

When using the C programming interface, the header data is transferred partly as a call parameter to the log3open(5) and log3write(5) subroutines and is partly supplemented by these subroutines.

In order to also transfer the header information to Logging V3.0 when using the syslog(3C) interface or when using the logger(1-ucb) command, the header data prefixes the message text in this case. The byte positions should be observed as follows here:

1	–	6	Magic string: "LOG3.0"
7	–	16	Time in seconds since 1970
17	–	26	Component number
27	–	31	Error number
32	–	36	Error weight
37	–	41	Structure number
42	–	53	Release/version of component
54	–	69	Module name. The module name may on contain the ":" character).

The prefix is recognized by the syslogd(1M) daemon, converted to header information and passed on to Logging V3.0 using the C programming interface. Important note: This does not apply for logger(1) !

EXAMPLES

C interface (see also the manual *Diagnostics and Troubleshooting*, chap. "Passing Sylog data"):

```
(void) time(&now)
(void) sprintf(head, "LOG3.0%-10lu%-10lu%-5u%-5u%-5u%-12.12s%-16.16s",
    now, compno, errno, level, structid, rel_ver, ident);
syslog(LOG_WARNING, "%smessage text and value %d", head, value);
```

Command interface:

```
compno=1080009
errno=131
level=6
structid=0
rel_ver=""
ident="rdd"
head=`printf "LOG3.0.........%-10lu%-5u%-5u%-5u%-12.12s%-16.16s"  \
    $compno $errno $level $structid "$rel_ver" "$ident"`
program=$0
/usr/ucb/logger -p 6 -i -t $program "${head}message texts and values"
```

SEE ALSO

logger(1-ucb), syslogd(1M), syslog(3C), syslog.conf(4), log3(5), log3open(5), log3write(5).

System Administrator's Reference Manual

log3memcpy(5)

NAME

`log3memcpy` - copy log data to the log buffer

SYNOPSIS

```
int log3memcpy(data, size)
char *data;           /* Address of memory area to be logged   */
int size;             /* Length of this memory area (in bytes) */
```

DESCRIPTION

The `log3memcpy` function copies log data to the buffer.

Return value:

`≥0` Number of bytes transferred to the log buffer.

`−1` Pipe not open or log buffer full.

If there is insufficient space in the log buffer due to the length of the log data, the log data is truncated.

SEE ALSO

`log3(5)`, `log3close(5)`, `log3open(5)`, `log3printf(5)`, `log3write(5)`.

System Administrator's Reference Manual

log3open(5)

NAME

`log3open` - set up communication connection

SYNOPSIS

```
int log3open(compno, rel_ver, usrtxt)
u_long compno; /* Unique component no. (assigned by FSC) */
char *rel_ver; /* Release/version of component, max. 12 char. */
char *usrtxt; /* Module name */
```

DESCRIPTION

The `log3open` function sets up the connection to the log server.

Return value:

`0` Pipe to log server open.

`-1` Error. `lg_errno` contains the error type. `lg_errno` and error return values are defined in `logging.h`.

In the case of repeated calls, `log3close()` is first executed internally.

NOTES

The module name `usrtxt` may only comprise digits, letters and "_" (in particular it must not contain the ":" character).

SEE ALSO

`log3(5)`, `log3close(5)`, `log3memcpy(5)`, `log3printf(5)`, `log3write(5)`.

System Administrator's Reference Manual

log3printf(5)

NAME

`log3printf` - transfer formatted log data to the log buffer

SYNOPSIS

```
int log3printf(format, args)
char *format;                   /* Format string as with printf() */
va_list args;                   /* Arguments to format string     */
```

DESCRIPTION

The `log3printf` function enters formatted log data in a static buffer.

Return value:

`≥0` Number of transferred bytes in the log buffer.

`−1` Pipe not open or log buffer full.

If there is insufficient space in the log buffer due to the length of the log data, the log data is truncated.

SEE ALSO

`log3(5)`, `log3close(5)`, `log3memcpy(5)`, `log3open(5)`, `log3write(5)`.

System Administrator's Reference Manual

log3write(5)

NAME

`log3write -` transfer log data to the log server

SYNOPSIS

```
int log3write(erno, level, structid)
u_short erno;                                /* Error number      */
u_short level;                              /* Severity of error  */
u_short structid;         /* Structure ID for subsequent data    */
/* 0 = no structure ID necessary  */
```

DESCRIPTION

The `log3write` function transfers the log data from the log buffer to the log server.

Return value:

`0` Log data were sent to the log server.

`–1` Error. `lg_errno` contains the error type. `lg_errno` and error return values are defined in `logging.h`.

If errors occurred when writing (`errno=EBADF`, `EPIPE`, `EAGAIN`), an attempt is made to reopen the pipe. This process is repeated a maximum of five times.

The pipe is always closed before an error exit.

SEE ALSO

`log3(5)`, `log3close(5)`, `log3memcpy(5)`, `log3open(5)`, `log3printf(5)`.

System Administrator's Reference Manual

man(5)

NAME

`man` - macros for formatting entries in a manual (man pages)

SYNOPSIS

`nroff –man` *file* ...

`troff –man` [`–rs1`] *file* ...

DESCRIPTION

These `troff(1M)` macros are used to lay out the format of the entries of a manual. These macros are used by the `man(1)` command.

The default page size is 8.5" × 11", with a 6.5" × 10" text area; the `–rs1` option reduces these dimensions to 6" × 9" and 4.75" × 8.375", respectively; this option [which is *not* effective in `nroff(1M)`] also reduces the default type size from 10-point to 9-point, and the vertical line spacing from 12-point to 10-point. The `–rV2` option may be used to set certain parameters to values appropriate for certain Versatec printers: it sets the line length to 82 characters, the page length to 84 lines, and it inhibits underlining; this option should not be confused with the `–Tvp` option of the `man(1)` command, which is available at some UNIX system sites.

Any *text* argument below may be one to six "words". Double quotes ("") may be used to include blanks in a "word". If *text* is empty, the special treatment is applied to the next line that contains text to be printed. For example, `.I` may be used to italicize a whole line, or `.SM` followed by `.B` to make small bold text. By default, hyphenation is turned off for `nroff(1M)`, but remains on for `troff(1M)`.

Type font and size are reset to default values before each paragraph and after processing font- and size-setting macros, e.g. `.I`, `.RB`, `.SM`. Tab stops are neither used nor set by any macro except `.DT` and `.TH`.

Default units for indents *in* are `ens`. When *in* is omitted, the previous indent is used. This remembered indent is set to its default value (7.2ens in `troff`, 5ens in `nroff`; this corresponds to 0.5" in the default page size) by `.TH`, `.P`, and `.RS`, and restored by `.RE`.

`.TH` *t s c n*
: Set the title and entry heading; *t* is the title, *s* is the section number, *c* is extra commentary, e.g. "local", *n* is new manual name. Invokes `.DT` (see below).

`.SH` *text*
: Place subhead *text*, e.g. `SYNOPSIS`, here.

`.SS` *text*
: Place sub-subhead *text*, e.g. `Attributes`, here.

`.B` *text*
: Make *text* bold.

`.I` *text*
: Make *text* italic.

`.SM` *text*
: Make *text* 1 point smaller than default point size.

`.RI` *a b*
: Concatenate roman *a* with italic *b*, and alternate these two fonts for up to six arguments. Similar macros alternate between any two of roman, italic, and bold: `.IR .RB .BR .IB .BI`

`.P`
: Begin a paragraph with normal font, point size, and indent.

`.PP`
: Synonym for `.P`.

`.HP` *in*
: Begin paragraph with hanging indent.

`.TP` *in*

Begin indented paragraph with hanging tag. The next line that contains text to be printed is taken as the tag. If the tag does not fit, it is printed on a separate line.

`.IP` *t in*
Same as `.TP` *in* with tag *t*; often used to get an indented paragraph without a tag.

`.RS` *in*
Increase relative indent (initially zero). Indent all output an extra *in* units from the current left margin.

`.RE` *k*
Return to the *k*th relative indent level (initially, *k*=1; *k*=0 is equivalent to *k*=1); if *k* is omitted, return to the most recent lower indent level.

`.PM` *m y*
Produces proprietary markings; where *m* may be `P` for Proprietary, `RS` for Restricted, `RG` for Registered, `CR` for Copyright, `UW` for Unpublished Work, or `CP` for See Cover Page; where *y* is the year used for the copyright date for notices `CR` and `UW` (default is current year).

`.DT` Restore default tab settings (every 7.2ens in `troff`, 5ens in `nroff`).

`.PD` *v*
Set the interparagraph distance to *v* vertical spaces. If *v* is omitted, set the interparagraph distance to the default value (0.4v in `troff`, 1v in `nroff`).

The following *strings* are defined:

`\*R` ® in `troff`, (`Reg.`) in `nroff`.

`\*S` Change to default point size.

`\*(Tm`
Trademark indicator.

The following *number registers* are given default values by `.TH`:

`IN` Left margin indent relative to subheads (default is 7.2ens in `troff`, 5ens in `nroff`).

`LL` Line length including `IN`.

`PD` Current interparagraph distance.

NOTES

In addition to the macros, strings, and number registers mentioned above, a number of *internal* macros, strings, and number registers are also defined. Except for names predefined by `troff`(1M) and number registers `d`, `m`, and `y`, all such internal names are of the form *XA*, where *X* is one of),], and }, and *A* stands for any alphanumeric character.

If a manual entry needs to be preprocessed by `eqn`(1M) [or `neqn`(1M)], and/or `tbl`(1M) , it must begin with a special line [described in `man`(1)], causing the `man` command to invoke the appropriate preprocessor(s).

The programs that prepare the Table of Contents and the Permuted Index for this Manual assume the `NAME` section of each entry consists of a single line of input that has the following format:

name[, *name*, *name* ...] \– *explanatory text*

The macro package increases the inter-word spaces (to eliminate ambiguity) in the `SYNOPSIS` section of each entry.

The macro package itself uses only the roman font (so that one can replace, for example, the bold font by the constant-width font (`CW`). Of course, if the input text of an entry contains requests for other fonts (e.g. `.I`, `.RB`, `\fI`), the corresponding fonts must be mounted.

NOTE

If the argument to `.TH` contains *any* blanks and is *not* enclosed by double quotes (""), there will be strange irregular dots on the output.

FILES

```
/usr/lib/tmac/tmac.an
/usr/lib/macros/an
```

SEE ALSO

man(1), eqn(1M), nroff(1M), tbl(1M), tc(1M), troff(1M).

System Administrator's Reference Manual

mcolor(5)

NAME

`mcolor -` color and reverse video macro

SYNOPSIS

`troff –mcolor` [*option ...*] *file ...*

DESCRIPTION

A macro package for color selection and reverse video printing on PostScript printers. The package is compatible with most existing macro packages and includes the following macro:

`.CL` *color text*

Prints *text* in *color*. No arguments restore the default color (black). If *text* is omitted, the selected *color* remains in effect until another *color* is selected. If two arguments are given, the *text* is printed in *color* and then the default color is restored.

Both the text and background color can be selected. A *color* argument of "color1 on color2" prints text in color1 on a background in color2.

Named colors must be listed in the colordict dictionary in file `/usr/lib/lp/postscript/color.ps`.

FILES

`/usr/lib/tmac/tmac.color`

`/usr/lib/lp/postscript/color.ps`

SEE ALSO

`dpost`(1M), `troff`(1M).

System Administrator's Reference Manual

me(5)

NAME

`me` - macros for formatting papers

SYNOPSIS

`nroff –me` [*options*] *filename ...*

`troff –me` [*options*] *filename ...*

DESCRIPTION

This package of `nroff` and `troff` macro definitions provides a canned formatting facility for technical papers in various formats. When producing 2-column output on a terminal, filter the output through `col(1)`.

The macro requests are defined below. Many `nroff` and `troff` requests are unsafe in conjunction with this package, however these requests may be used with impunity after the first `.pp`:

`.bp`	begin new page
`.br`	break output line here
`.sp` *n*	insert *n* spacing lines
`.ls` *n*	(line spacing) *n*=1 single, *n*=2 double space
`.na`	no alignment of right margin
`.ce` *n*	center next *n* lines
`.ul` *n*	underline next *n* lines
`.sz` +*n*	add *n* to point size

Output of the `eqn(1M)`, `neqn(1M)`, and `tbl(1M)` preprocessors for equations and tables is acceptable as input.

REQUESTS

In the following list, "initialization" refers to the first `.pp, .lp, .ip, .np, .sh`, or `.uh` macro. This list is incomplete.

Request	Cause Break	If no Argument	Explanation
`.(c`	–	yes	Begin centered block.
`.(d`	–	no	Begin delayed text.
`.(f`	–	no	Begin footnote.
`.(l`	–	yes	Begin list.
`.(q`	–	yes	Begin major quote.
`.(x`*x*	–	no	Begin indexed item in index *x*.
`.(z`	–	no	Begin floating keep.
`.)c`	–	yes	End centered block.

.)d	–	yes	End delayed text.
.)f	–	yes	End footnote.
.)l	–	yes	End list.
.)q	–	yes	End major quote.
.)x	–	yes	End index item.
.)z	–	yes	End floating keep.
.++ *m H*	–	no	Define paper section. *m* defines the part c B (bibliography), RC (chapters renumbere
.+c *T*	–	yes	Begin chapter (or appendix, etc. as set by
.1c	1	yes	One column format on a new page.
.2c	1	yes	Two column format.
.EN	–	yes	Space after equation produced by eqn or
.EQ *x y*	–	yes	Precede equation; break out and add spa equation, or C to center the equation.
.GE	–	yes	End gremlin picture.
.GS	–	yes	Begin gremlin picture.
.PE	–	yes	End pic picture.
.PS	–	yes	Begin pic picture.
.TE	–	yes	End table.
.TH	–	yes	End heading section of table.
.TS *x*	–	yes	Begin table; if *x* is H table has repeated he
.ac *A N*	–	no	Set up for ACM style output. *A* is the Auth
.b *x*	no	no	Print *x* in boldface; if no argument switch t
.ba +*n*	0	yes	Augments the base indent by *n*. This inde
.bc	no	yes	Begin new column.
.bi *x*	no	no	Print *x* in bold italics (nofill only).
.bu	–	yes	Begin bulleted paragraph.
.bx *x*	no	no	Print *x* in a box (nofill only).
.ef *'x'y'z*	"""	no	Set even footer to *x y z*.
.eh *'x'y'z*	"""	no	Set even header to *x y z*.
.fo *'x'y'z*	"""	no	Set footer to *x y z*.
.hx	–	no	Suppress headers and footers on next pa
.he *'x'y'z*	"""	no	Set header to *x y z*.
.hl	–	yes	Draw a horizontal line.
.i *x*	no	no	Italicize *x*; if *x* missing, italic text follows.
.ip *x y*	no	yes	Start indented paragraph, with hanging ta
.lp	yes	yes	Start left-blocked paragraph.

.lo	–	no	Read in a file of local macros of the form
.np	1	yes	Start numbered paragraph.
.of *'x'y'z*	"""	no	Set odd footer to *x y z*.
.oh *'x'y'z*	"""	no	Set odd header to *x y z*.
.pd	–	yes	Print delayed text.
.pp	no	yes	Begin paragraph. First line indented.
.r	yes	no	Roman text follows.
.re	–	no	Reset tabs to default values.
.sc	no	no	Read in a file of special characters and di
.sh *n x*	–	yes	Section head follows, font automatically b
.sk	no	no	Leave the next page blank. Only one pag
.sm *x*	–	no	Set *x* in a smaller pointsize.
.sz +*n*	10p	no	Augment the point size by *n* points.
.th	no	no	Produce the paper in thesis format. Must
.tp	no	yes	Begin title page.
.u *x*	–	no	Underline argument (even in troff). (No
.uh	–	yes	Like .sh but unnumbered.
.xp *x*	–	no	Print index *x*.

FILES

/usr/lib/tmac/tmac.e
/usr/ucblib/doctools/tmac/tmac.e

SEE ALSO

refer(1), eqn(1M), nroff(1M), troff(1M), tbl(1M).

System Administrator's Reference Manual

mm(5)

NAME

mm - the mm macro package for formatting documents

SYNOPSIS

mm [*option* ...] *file* ...

nroff [*option* ...] –mm *file* ...

mmt [*option* ...] *file* ...

troff [*option* ...] –mm *file* ...

DESCRIPTION

This package provides a formatting capability for a very wide variety of documents. The manner in which you type and edit a document is essentially independent of whether the document is to be eventually formatted at a terminal or is to be phototypeset. See the **REFERENCE** below for further details.

FILES

/usr/lib/tmac/tmac.m
: pointer to the mm package

/usr/lib/macros/mmn
: macro definitions for nroff

/usr/lib/macros/mmt
: macro definitions for troff

/usr/lib/macros/strings.*
: company-specific string definitions

SEE ALSO

mm(1M), mmt(1M), nroff(1M), troff(1M).

REFERENCE

D. W. Smith, J. R. Mashey, *mm — Memorandum Macros*

System Administrator's Reference Manual

mpictures(5)

NAME

`mpictures -` picture inclusion macros

SYNOPSIS

`troff –mpictures` [*options*] *file* ...

DESCRIPTION

A macro package used to include PostScript pictures in `troff` documents. The package is compatible with many existing `troff` macro packages and includes the following three macros:

`.BP` *file height width position offset flags label*

Places the picture *file* in the space set aside by *height*, *width*, *position*, and *offset*, which together define and position the picture frame. The macro arguments are:

file Pathname of a PostScript picture *file*. Appending (*n*) to *file* selects page number *n* from a multipage picture *file*. By default the first page in *file* is selected.

height

Vertical extent of the frame. The default is 3.0i.

width

Horizontal extent of the frame. The default is the current length of a line of text.

position

One of `l`, `c`, or `r` used to align the left, center, or right of the frame with the corresponding position on the current line of text. The default is `l`.

offset

Moves the frame right (positive) or left (negative) from the selected *position*. The default is 0i.

flags

A string built from one or more of the following:

`a[d]` rotate the picture clockwise `d` degrees. If `d` is omitted 90 degrees is added to the current angle, which starts at zero.

`o` outline the picture with a box

`s` freely scale both picture dimensions

`w` white out the area to be occupied by the picture

`l` attach picture to the left

`r` right

`t` top

`b` or bottom of the frame

label

Place *label* 1.5 vertical units below the frame.

If there's room `.BP` fills text around the frame. Everything destined for either side of the frame first goes into a diversion and only reappears when the accumulated text sweeps past the trap set by `.BP` or when the diversion is explicitly closed by the `.EP` macro (below).

Null arguments, represented by "", are replaced by the defaults as noted above.

`.PI` *file height,width,yoffset,xoffset flags*

A low level macro used by `.BP`. It can help if you're trying to do things that `.BP` won't allow or doesn't do well. The two arguments not already described are:

xoffset

Moves the frame right (positive) or left (negative) from the left margin. The default is 0i.

yoffset
Moves the frame down (positive) or up (negative) from the current baseline. The default is 0i.

The second argument is a comma-separated list of four numbers, and although defaults are available, supplying values for all four numbers is recommended.

`.EP` Ends a picture started by `.BP`. An explicit `.EP` call is not often required. Instead `.EP` is usually called by springing the trap set by `.BP` at the bottom of each frame.

Much of what's done depends on file structuring comments commonly found in PostScript files. If the comments needed to isolate a particular page are missing the entire *file* is included. If a `%%BoundingBox` comment is missing the picture is assumed to fill an 8.5×11 inch page. A picture *file* that can't be read when the `troff` postprocessor runs is replaced by white space. Nothing done in `.BP` or `.PI` guarantees the picture hasn't been placed off the page. All dimensions should be explicitly given in inches.

NOTES

A picture and associated text can silently disappear if the diversion trap set by `.BP` isn't reached. Including a call to `.EP` at the end of the paper should recover whatever appears to be missing.

Macros in other packages occasionally break the adjustments made to the line length and indent when text is being placed around a picture.

A missing or improper `%%BoundingBox` comment often explains why a picture doesn't properly fill the space that's been set aside.

FILES

`/usr/lib/tmac/tmac.pictures`

SEE ALSO

`dpost`(1M), `picpack`(1M), `troff`(1M).

System Administrator's Reference Manual

mptx(5)

NAME

`mptx` - the macro package for formatting a permuted index

SYNOPSIS

`nroff –mptx` [*option ...*] *file ...* | *printer*

`troff –mptx` [*option ...*] *file ...* | *typesetter*

DESCRIPTION

This package provides a definition for the `.xx` macro used for formatting a permuted index as produced by `ptx`(1M) . This package does not provide any other formatting capabilities such as headers and footers. If these or other capabilities are required, the `mptx` macro package may be used in conjunction with the `mm` macro package. In this case, the `–mptx` option must be invoked after the `–mm` call. For example,

`nroff –mm –mptx` *file ...* | *printer* or `mmt –mptx` *file ...* | *typesetter*

FILES

`/usr/lib/tmac/tmac.ptx`
pointer to the macro package

`/usr/lib/macros/ptx`
macro package

SEE ALSO

`nroff`(1M) , `ptx`(1M) , `troff`(1M) .

System Administrator's Reference Manual

ms(5)

NAME

`ms` - macros for formatting manuscripts

SYNOPSIS

`nroff –ms` [*options*] *file ...*

`troff –ms` [*options*] *file ...*

DESCRIPTION

This package of `nroff(1M)` and `troff(1M)` macro definitions provides a canned formatting facility for technical papers in various formats. When producing 2-column output on a terminal, filter the output through `col(1)`.

The macro requests are defined below. Many `nroff` and `troff` requests are unsafe in conjunction with this package, but the following requests may be used with impunity after the first `.PP`: `.bp`, `.br`, `.sp`, `.ls`, `.na`.

Output of the `eqn(1M)`, `neqn(1M)`, `tbl(1M)`, and `pic(1M)` preprocessors for equations, tables, and pictures is acceptable as input.

Diacritical marks may be applied to letters, as in these examples:

*`e	*`a	*'e	*^e	*^o
è	à	é	ê	ô

REQUESTS

Request	Initial Value	Cause	Explanation
`.1C`	yes	yes	One column format on a new page.
`.2C`	no	yes	Two column format.
`.AB`	no	yes	Begin abstract.
`.AE`	–	yes	End abstract.
`.AI`	no	yes	Author's institution follows. Suppressed in
`.AT`	no	yes	Print "Attached" and turn off line filling.
`.AU` *x y*	no	yes	Author's name follows. *x* is location and *y*
`.B` *x y*	no	no	Print *x* in boldface, append *y*; if no argume
`.B1`	no	yes	Begin text to be enclosed in a box.
`.B2`	no	yes	End boxed text.
`.BI` *x y*	no	no	Print *x* in bold italic and append *y*; if no ar
`.BT`	date	no	Bottom title, automatically invoked at foot
`.BX` *x*	no	no	Print *x* in a box.
`.CW` *x y*	no	no	Constant width font for *x*, append *y*; if no a
`.CT`	no	yes	Print "Copies to" and turn off line filling.
`.DA` *x*	nroff	no	"Date line" at bottom of page is *x*. Default
`.DE`	–	yes	End displayed text. Implies `.KE`.

.DS *x*	no	yes	Start of displayed text, to appear verbatim
.EG	no	–	Print document in BTL format for "Engine
.EN	–	yes	Space after equation produced by neqn
.EQ *x y*	–	yes	Display equation. Equation number is *y*. C
.FE	–	yes	End footnote.
.FP *x*	–	no	Set font positions for a family, e.g. .FP p
.FS	no	no	Start footnote. The note will be moved to
.HO	–	no	"Bell Laboratories, Holmdel, New Jersey (
.I *x y*	no	no	Italicize *x*, append *y*; if no argument switcl
.IH	no	no	"Bell Laboratories, Naperville, Illinois 605
.IM	no	no	Print document in BTL format for an interr
.IP *x y*	no	yes	Start indented paragraph, with hanging ta
.KE	–	yes	End keep. Put kept text on next page if nc
.KF	no	yes	Start floating keep. If the kept text must b
.KS	no	yes	Start keeping following text.
.LG	no	no	Make letters larger.
.LP	yes	yes	Start left-blocked paragraph.
.LT	no	yes	Start a letter with today's date; address fo
.MF	–	–	Print document in BTL format for "Memor
.MH	–	no	"Bell Laboratories, Murray Hill, New Jerse
.MR	–	–	Print document in BTL format for "Memor
.ND *date*	10p	no	Use date supplied (if any) only in special
.NH *n*	–	yes	Same as .SH, with automatic section num
.NL	yes	no	Make letters normal size.
.PE	–	yes	End picture [see pic(1M)].
.PF	–	yes	End picture; restore vertical position.
.PP	no	yes	Begin paragraph. First line indented.
.PS *h w*	–	yes	Start picture; height and width in inches.
.PY	–	no	"Bell Laboratories, Piscataway, New Jers
.QE	–	yes	End quoted material.
.QP	–	yes	Begin quoted paragraph (indent both mar
.QS	–	yes	Begin quoted material (indent both margi
.R	yes	no	Roman text follows.
.RE	–	yes	End relative indent level.
.RP	no	–	Cover sheet and first page for released p
.RS	–	yes	Start level of relative indentation from whi

.SG *x*	no	yes	Insert signature(s) of author(s), ignored e
.SH	–	yes	Section head follows, font automatically b
.SM	no	no	Make letters smaller.
.TA *x* ...	5 ...	no	Set tabs in ens. Default is 5 10 15 ...
.TE	–	yes	End table [see tbl(1M)].
.TH	–	yes	End heading section of table.
.TL	no	yes	Title follows.
.TM *x* ...	no	–	Print document in BTL technical memora requests.
.TR *x*	–	–	Print in BTL technical report format; repor
.TS *x*	–	yes	Begin table; if *x* is H table heading is repe
.UL *x*	–	no	Underline argument.
.UX *y z*	–	no	"*z*UNIX*y*"; first use gives registered traden
.WH	–	no	"Bell Laboratories, Whippany, New Jerse
.[	–	no	Begin reference.
.]	–	no	End reference.

FILES

/usr/lib/tmac/tmac.s

SEE ALSO

eqn(1M), pic(1M), tbl(1M), troff(1M).

System Administrator's Reference Manual

mv(5)

NAME

`mv` - mv macro package for typesetting viewgraphs and slides

SYNOPSIS

`mvt` [`−a`] [*option ...*] *file ...* □
`troff` [`−a`] [`−rX1`] `−mv` [*option ...*] *file ...*

DESCRIPTION

This package makes it easy to typeset viewgraphs and projection slides in a variety of sizes. A few macros (briefly described below) accomplish most of the formatting tasks needed in making transparencies. All of the facilities of `troff`(1M), `eqn`(1M), `tbl`(1M), `pic`(1M), and `grap`(1M) are available for more difficult tasks.

The output can be previewed on most terminals, and, in particular, on the TEKTRONIX 4014. For this device, specify the `−rX1` option (this option is automatically specified by the `mvt` command when that command is invoked with the `−D` option). To preview output on other terminals, specify the `−a` option.

The available macros are:

`.VS` [*n*] [*i*] [*d*]
Foil-start macro; foil size is to be 7" × 7"; *n* is the foil number, *i* is the foil identification, *d* is the date; the foil-start macro resets all parameters (indent, point size, etc.) to initial default values, except for the values of *i* and *d* arguments inherited from a previous foil-start macro; it also invokes the `.A` macro (see below).

The naming convention for this and the following eight macros is that the first character of the name (`V` or `S`) distinguishes between viewgraphs and slides, respectively, while the second character indicates whether the foil is square (`S`), small wide (`w`), small high (`h`), big wide (`W`), or big high (`H`). Slides are "skinnier" than the corresponding viewgraphs: the ratio of the longer dimension to the shorter one is larger for slides than for viewgraphs. As a result, slide foils can be used for viewgraphs, but not vice versa; on the other hand, viewgraphs can accommodate a bit more text.

`.Vw` [*n*] [*i*] [*d*]
Same as `.VS`, except that foil size is 7" wide × 5" high.

`.Vh` [*n*] [*i*] [*d*]
Same as `.VS`, except that foil size is 5" × 7".

`.VW` [*n*] [*i*] [*d*]
Same as `.VS`, except that foil size is 7" × 5.4".

`.VH` [*n*] [*i*] [*d*]
Same as `.VS`, except that foil size is 7" × 9".

`.Sw` [*n*] [*i*] [*d*]
Same as `.VS`, except that foil size is 7" × 5".

`.Sh` [*n*] [*i*] [*d*]
Same as `.VS`, except that foil size is 5" × 7".

`.SW` [*n*] [*i*] [*d*]
Same as `.VS`, except that foil size is 7" × 5.4".

`.SH` [*n*] [*i*] [*d*]
Same as `.VS`, except that foil size is 7" × 9".

`.A` [*x*]
Place text that follows at the first indentation level (left margin); the presence of *x* suppresses the ½ line spacing from the preceding text.

`.B` [*m* [*s*]]
Place text that follows at the second indentation level; text is preceded by a mark; *m* is the mark (default is a large bullet); *s* is the increment or decrement to the point size of the mark with respect to the *prevailing* point size (default is 0); if *s* is 100, it causes the point size of the mark to be the same as that of the *default* mark.

`.C` [*m* [*s*]]
Same as `.B`, but for the third indentation level; default mark is a dash.

`.D` [*m* [*s*]]
Same as `.B`, but for the fourth indentation level; default mark is a small bullet.

`.T` *string*
string is printed as an over-size, centered title.

`.I` [*in*] [*a* [*x*]]
Change the current text indent (does not affect titles); *in* is the indent (in inches unless dimensioned, default is 0); if *in* is signed, it is an increment or decrement; the presence of *a* invokes the `.A` macro (see below) and passes *x* (if any) to it.

`.S` [*p*] [*l*]
Set the point size and line length; *p* is the point size (default is "previous"); if *p* is 100, the point size reverts to the *initial* default for the current foil-start macro; if *p* is signed, it is an increment or decrement (default is 18 for `.VS`, `.VH`, and `.SH`, and 14 for the other foil-start macros); *l* is the line length (in inches unless dimensioned; default is 4.2" for `.Vh`, 3.8" for `.Sh`, 5" for `.SH`, and 6" for the other foil-start macros).

`.DF` *n f* [*n f* ...]
Define font positions; may not appear within a foil's input text (i.e., it may only appear after all the input text for a foil, but before the next foil-start macro); *n* is the position of font *f*; up to four *n*□*f* pairs may be specified; the first font named becomes the *prevailing* font; the initial setting is (`H` is a synonym for `G`):
`.DF 1 H 2 I 3 B 4 S`

`.DV` [*a*] [*b*] [*c*] [*d*]
Alter the vertical spacing between indentation levels; *a* is the spacing for `.A`, *b* is for `.B`, *c* is for `.C`, and *d* is for `.D`; all non-null arguments must be dimensioned; null arguments leave the corresponding spacing unaffected; initial setting is: `.DV .5v .5v .5v 0v`

`.U` *str1* [*str2*]
Underline *str1* and concatenate *str2* (if any) to it.

The last four macros in the above list do not cause a break; the `.I` macro causes a break only if it is invoked with more than one argument; all other macros force a break.

The macro package also recognizes the following uppercase synonyms for the corresponding lowercase `troff` requests:

`.AD .BR .CE .FI .HY .NA .NF .NH .NX .SO .SP .TA .TI`

The `Tm` string produces the trademark symbol.

See the reference cited below for further details.

NOTE

The `.VW` and `.SW` foils are meant to be 9" wide by 7" high, but because the typesetter paper is generally only 8" wide, they are printed 7" wide by 5.4" high and have to be enlarged by a factor of 9/7 before use as viewgraphs; this makes them less than totally useful.

FILES

`/usr/lib/tmac/tmac.v`
`/usr/lib/macros/vmca`

SEE ALSO

`mmt`(1M) , `troff`(1M) .

REFERENCE

T. A. Dolotta, *A Macro Package for View Graphs and Slides*

System Administrator's Reference Manual

nterm(5)

NAME

`nterm` - terminal driving tables for nroff

DESCRIPTION

`nroff`(1M) uses driving tables to customize its output for various types of output devices, such as printing terminals, special word-processing terminals (such as Diablo, Qume, or NEC Spinwriter mechanisms), or special output filter programs. These driving tables are written as ASCII files, and are installed in `/usr/lib/nterm/tab.`*name*, where *name* is the name for that terminal type as given in `nroff`(1M).

The first line of a driving table should contain the name of the terminal: simply a string with no embedded white space. "White space" means any combination of spaces, tabs and newlines. The next part of the driver table is structured as follows:

`bset`	*[*integer]	(not supported in all versions of `nroff`)
`breset`	*[*integer]	(not supported in all versions of `nroff`)
`Hor`	*[*integer]	
`Vert`	*[*integer]	
`Newline`	*[*integer]	
`Char`	*[*integer]	
`Em`	*[*integer]	
`Halfline`	*[*integer]	
`Adj`	*[*integer]	
`twinit`	*[*character string]	
`twrest`	*[*character string]	
`twnl`	*[*character string]	
`hlr`	*[*character string]	
`hlf`	*[*character string]	
`flr`	*[*character string]	
`bdon`	*[*character string]	
`bdoff`	*[*character string]	
`iton`	*[*character string]	
`itoff`	*[*character string]	
`ploton`	*[*character string]	
`plotoff`	*[*character string]	
`up`	*[*character string]	
`down`	*[*character string]	
`right`	*[*character string]	
`left`	*[*character string]	

The meanings of these fields are as follows:

`bset` bits to set in the `c_oflag` field of the `termio` structure before output.

`breset`
bits to reset in the `c_oflag` field of the `termio` structure before output.

`Hor` horizontal resolution in units of 1/240 of an inch.

`Vert` vertical resolution in units of 1/240 of an inch.

`Newline`
space moved by a newline (linefeed) character in units of 1/240 of an inch.

`Char` quantum of character sizes, in units of 1/240 of an inch (i.e. a character is a multiple of `Char` units wide)

`Em` size of an em in units of 1/240 of an inch.

`Halfline`
space moved by a half-linefeed (or half-reverse-linefeed) character in units in 1/240 of an inch.

`Adj` quantum of white space, in 1/240 of an inch (i.e. white spaces are a multiple of `Adj` units wide)
Note: if this is less than the size of the space character, `nroff` will output fractional spaces using plot mode. Also, if the `–e` switch to `nroff` is used, `Adj` is set equal to `Hor` by `nroff`.

`twinit`
sequence of characters used to initialize the terminal in a mode suitable for `nroff`.

`twrest`
sequence of characters used to restore the terminal to normal mode.

`twnl` sequence of characters used to move down one line.

`hlr` sequence of characters used to move up one-half line.

`hlf` sequence of characters used to move down one-half line.

`flr` sequence of characters used to move up one line.

`bdon` sequence of characters used to turn on hardware boldface mode, if any.

`bdoff` sequence of characters used to turn off hardware boldface mode, if any.

`iton` sequence of characters used to turn on hardware italics mode, if any.

`itoff` sequence of characters used to turn off hardware italics mode, if any.

`ploton`
sequence of characters used to turn on hardware plot mode (for Diablo type mechanisms), if any.

`plotoff`
sequence of characters used to turn off hardware plot mode (for Diablo type mechanisms), if any.

`up` sequence of characters used to move up one resolution unit (Vert) in plot mode, if any.

`down` sequence of characters used to move down one resolution unit (Vert) in plot mode, if any.

`right` sequence of characters used to move right one resolution unit (Hor) in plot mode, if any.

`left` sequence of characters used to move left one resolution unit (Hor) in plot mode, if any.

This part of the driving table is fixed format, and you cannot change the order of entries. You should put entries on separate lines, and these lines should contain exactly two fields (no comments allowed) separated by white space. For example,

```
bset      0
breset    0
Hor       24
```

and so on.

Follow this first part of the driving table with a line containing the word `charset`, and then specify a table of special characters that you want to include. That is, specify all the non-ASCII characters that `nroff`(1M) knows by two character names, such as `\(hy`. If `nroff` does not find the word `charset` where it expects to,

it will abort with an error message.

Each definition in the part after `charset` occupies one line, and has the following format:

```
chname width output
```

where `chname` is the two-character name (e.g. `\(dg` for dagger) of the special character, `width` is its width in ems, and `output` is the string of characters and escape sequences to send to the terminal to produce the special character.

If any field in the `charset` part of the driving table does not pertain to the output device, you may give that particular sequence as a null string, or leave out the entry. Special characters that do not have a definition in this file are ignored on output by `nroff(1M)` .

The following escapes may be used in the definition of the character output:

Escape	Function
`\b`	backspace
`\t`	tab
`\r`	return (no line feed)
`\n`	line return (new line)

You may put the `charset` definitions in any order, so it is possible to speed up `nroff` by putting the most frequently used characters first. For example,

```
charset
em      1    -
hy      1    -
\-      1    -
bu      1    +\bo
```

and so on.

The best way to create a terminal table for a new device is to take an existing terminal table and edit it to suit your needs. Once you create such a file, put it in the directory `/usr/lib/nterm`, and give it the name `tab.`*xyz* where *xyz* is the name of the terminal and the name that you pass `nroff` via the `–T` option (for example, `nroff –Txyz`).

NOTES

Normal 8-bit input is an encoding of local motions interpreted by `nroff`. For example, octal codes 000 through 177 represent ordinary 7-bit ASCII. Some local motions (for octal codes 200 through 377) include: 200 means "right", 240 means "left", 300 means "down" and 340 means "up", any of which can be used with a 5-bit value (octal codes 001 through 037) to represent the magnitude of motion.

By default, `nroff` treats any octal value of 200 or greater as a local motion; however, if a definition includes a `%`, the following byte is output verbatim. To pass through an octal code for a particular device, uninterpreted by `nroff`, use the form `%\`*nnn* where *nnn* is the octal value (three digits).

FILES

`/usr/lib/nterm/tab.`*name* terminal files

SEE ALSO

`nroff(1M)` .

System Administrator's Reference Manual

roff-charset(5)

NAME

`roff-charset` - special characters for n/troff

DESCRIPTION

`nroff`(1M) and `troff`(1M) have a series of special characters, which are rewritten as `\`*c* or `\(`*cc* (*c* stands for a particular character) in the input text. These special characters include normal 7-bit ASCII characters [see `ascii`(5)], 8-bit metacharacters [see `meta-ascii`(5)], mathematical symbols, the entire Greek alphabet, and other miscellaneous characters.

Because of the wider range of (HP-LaserJet or PostScript) printer options, `troff` generally recognizes – depending on the version installed – far more special characters than `nroff`. Those special characters that are recognized by both `troff` and `nroff` are defined in the `/usr/lib/nterm/tab.`*name* file [see `nterm`(5)] under the "charset" entry (*name* is the name of an output device specified when `nroff` or `troff` is called).

The list below contains only those special characters that are valid for both `nroff` and `troff`. Because a simple 7-bit terminal is used as an output device for `nroff`, the characters generated are usually only useful as an emergency stopgap measure. In this regard, an LP printer produces far better results.

Input	troff-Output	nroff-Output	Name/Meaning
\-	-	-	current font minus
\e	\	\	representation of the actual escape character (standard: \)
\'	´	´	acute accent (see **NOTES** toward the end)
\`	`	`	grave accent
\(!=	≠	/	not equal
\(**	*	*	asterisk
\(+-	±	±	plus-minus
\(->	→	->	right arrow
\(<-	←	<-	left arrow
\(<=	≤	≤	less than or equal to
\(>=	≥	≥	greater than or equal to
\(==	≡	=	identically equal
\(~=	≃	=	approximately equal
\(12	½	1/2	one-half
\(14	¼	1/4	one-fourth
\(34	¾	3/4	three-fourth
\(Fi	ffi	ffi	ffi ligature
\(Fl	ffl	ffl	ffl ligature

Input	troff-Output	nroff-Output	Name/Meaning
\(ff	ff	ff	ff-Ligatur
\(fi	fi	fi	fi-Ligatur
\(fl	fl	fl	fl-Ligatur
\(aa	´	'	Accent aigu (see **NOTES** towards the end)
\(ap	~	~	approximates
\(br	|	|	box vertical rule
\(bu	•	o	bullet
\(bv	|	|	bold vertical
\(ca	∩	(^)	cap (intersection)
\(cf	^	^	circumflex
\(ci	○	0	circle
\(co	©	0	copyright
\(ct	¢	/	cent sign
\(cu	∪	U	cup (union)
\(da	↓	v	down arrow
\(dd	‡	=	double dagger
\(de	°	o	degree
\(dg	†	-	dagger
\(di	÷	:	divide
\(em	—	-	¾ em dash
\(eq	=	=	math equals
\(es	∅	0	empty set
\(fm	´	'	foot mark
\(ga	`	`	Accent grave
\(gr	∇	V	gradient
\(hy	-	-	hyphen
\(ib	⊆	(=	improper subset
\(if	∞	oo	infinity
\(ip	⊇	=)	improper superset
\(is	∫	,	integral sign
\(lb	⌊	(	left bottom
\(lc	⌈	|	left ceiling
\(lf	⌊	|	left floor

Input	troff-Output	nroff-Output	Name/ Meaning
\(lh	☜	<=	left hand
\(lk	{	|	left center of big curly bracket
\(lt	⎧	(	left top
\(mi	-	-	math minus
\(mo	∈	c	member of
\(mu	x	x	multiply
\(no	¬	-	not
\(or	|	|	or
\(pd	∂	6	partial derivative
\(pl	+	+	math plus
\(pt	∝	oc	proportional to
\(rb	⎭	)	right bottom
\(rc	⎫	|	right ceiling
\(rf	⌋	|	right floor
\(rg	®	o	registered
\(rh	☞	=>	right hand
\(rk	}	|	right center of big curly bracket
\(rn	‾	-	root en extender
\(rt	⎫	)	right top
\(ru	-	_	rule
\(sb	⊂	(=	subset of
\(sc	§	f	section
\(sl	/	/	slash
\(sp	⊃	=)	superset of
\(sq	□	[]	square
\(sr	√	\/	square root
\(ua	↑	^	up arrow
\(ul	_	_	underrule
\(*A	Α	A	Alpha
\(*B	Β	B	Beta
\(*G	Γ	|~	Gamma
\(*D	Δ	/\	Delta
\(*E	Ε	E	Epsilon

Input	troff-Output	nroff-Output	Name/ Meaning
\(*Z	Z	Z	Zeta
\(*Y	H	H	Eta
\(*H	Θ	0	Theta
\(*I	I	I	Iota
\(*K	K	K	Kappa
\(*L	Λ	/\	Lambda
\(*M	M	M	My
\(*N	N	N	Ny
\(*C	Ψ	-	Xi
\(*O	O	0	Omikron
\(*P	Π	| |	Pi
\(*R	P	P	Rho
\(*S	Σ	≥	Sigma
\(*T	T	T	Tau
\(*U	Y	Y	Ypsilon
\(*F	Φ	0	Phi
\(*X	X	X	Chi
\(*Q	Ψ	U	Psi
\(*W	Ω	()	Omega
\(*a	α	a	alpha
\(*b	β	B	beta
\(*g	γ	y	gamma
\(*d	δ	o	delta
\(*e	ε	c	epsilon
\(*z	ζ	L	zeta
\(*y	η	n	eta
\(*h	θ	0	theta
\(*i	ι	i	iota
\(*k	κ	k	kappa
\(*l	λ	>	lambda
\(*m	μ	u	my
\(*n	ν	v	ny

Input	troff-Output	nroff-Output	Name/Meaning
\(*c	ξ	3	xi
\(*o	ο	o	omikron
\(*p	π	n	pi
\(*r	ρ	p	rho
\(*s	σ	o	sigma
\(*ts	ς	s	terminal sigma (Ende-sigma)
\(*t	τ	t	tau
\(*u	υ	u	ypsilon
\(*f	φ	o	phi
\(*x	χ	x	chi
\(*q	ψ	u	psi
\(*w	ω	w	omega

The 8-bit character set (ISO 8859-1) [cf. meta-ascii(5)] can also be accessed using the following special characters, with a few notable exceptions.

Hex.	Dec.	Okt.	Graphic	Meaning	Input
A0	240	160	␣	NBSP, Non Breaking Space (blank as part of a word)	**missing**
A1	241	161	¡	Inverse exclamation mark	\(!d
A2	242	162	¢	U.S. cent symbol	\(ct
A3	243	163	£	British Pound symbol	\(ps
A4	244	164	¤	General currency symbol	\(cr
A5	245	165	¥	Japanes Yen symbol	\(yn
A6	246	166	¦	Broken vertical line	**missing**
A7	247	167	§	Section symbol	\(sc
A8	250	168	¨	Umlaut (diaeresis) accent	\(:a
A9	251	169	©	Copyright symbol	\(CO
AA	252	170	ª	Fiminine ordinal indicator	\(of
AB	253	171	«	Opening guilements (angle quotes)	\(<<
AC	254	172	¬	NOT sign	\(no
AD	255	173	H Y	SHY, Soft Hyphen (possible seperator in a word)	\(en
AE	256	174	®	Registered symbol	\(RG
AF	257	175	¯	Over line (high line, Macron)	\(rn
B0	260	176	°	Degree (ring)	\(de
B1	261	177	±	Plus/Minus sign	\(+-
B2	262	178	²	Superscript 2	**missing**
B3	263	179	³	Superscript 3	**missing**
B4	264	180	´	Acute accent	\(aa
B5	265	181	µ	Lowercase mu (micro sign)	\(*m
B6	266	182	¶	Pilcrow (paragraph sign)	\(pg
B7	267	183	·	Middle dot	\(pc
B8	270	184	¸	Cedilla	\(,a
B9	271	185	¹	Superscript 1	**missing**
BA	272	186	º	Masculine ordinal indicator	\(om
BB	273	187	»	Closing guilements (angle quotes)	\(>>
BC	274	188	¼	One fourth (one quarter)	\(14
BD	275	189	½	One half	\(12
BE	276	190	¾	Three fourth	\(34
BF	277	191	¿	Inverse question mark	\(?d

Hex.	Dec.	Okt.	Graphic	Meaning	Input
C0	300	192	À	Uppercase A grave accent	\(A`
C1	301	193	Á	Uppercase A acute accent	\(A'
C2	302	194	Â	Uppercase A circumflex	\(A^
C3	303	195	Ã	Uppercase A tilde	\(A~
C4	304	196	Ä	Uppercase A umlaut or diaeresis	\(^A
C5	305	197	Å	Uppercase A degree	\(Ao
C6	306	198	Æ	Uppercase AE ligature	\(AE
C7	307	199	Ç	Uppercase C cedilla	\(Cc
C8	310	200	È	Uppercase E grave accent	\(E`
C9	311	201	É	Uppercase E acute accent	\(E'
CA	312	202	Ê	Uppercase E circumflex	\(E^
CB	313	203	Ë	Uppercase E umlaut or diaeresis	\(E:
CC	314	204	Ì	Uppercase I grave accent	\(I`
CD	315	205	Í	Uppercase I acute accent	\(I'
CE	316	206	Î	Uppercase I circumflex	\(I^
CF	317	207	Ï	Uppercase I umlaut or diaeresis	\(I:
D0	320	208	Ð	Uppercase Icelandic ETH	**missing**
D1	321	209	Ñ	Uppercase N tilde	\(N~
D2	322	210	Ò	Uppercase O grave accent	\(O`
D3	323	211	Ó	Uppercase O acute accent	\(O'
D4	324	212	Ô	Uppercase O circumflex	\(O^
D5	325	213	Õ	Uppercase O tilde	\(O~
D6	326	214	Ö	Uppercase O umlaut or diaeresis	\(^O
D7	327	215	×	Multiply sign	\(mu
D8	330	216	Ø	Oppercase O crossbar	\(O/
D9	331	217	Ù	Uppercase U grave accent	\(U`
DA	332	218	Ú	Uppercase U acute accent	\(U'
DB	333	219	Û	Uppercase U circumflex	\(U^
DC	334	220	Ü	Uppercase U umlaut or diaeresis	\(° U
DD	335	221	Ý	Uppercase Y acute accent	**missing**
DE	336	222	Þ	Uppercase Icelandic Thorn	**missing**
DF	337	223	ß	Sharp s	\(^s
E1	340	224	à	Lowercase a grave accent	\(a`
E1	341	225	á	Lowercase a acute accent	\(a'
E2	342	226	â	Lowercase a circumflex	\(a^
E3	343	227	ã	Lowercase a tilde	\(a~
E4	344	228	ä	Lowercase a umlaut or diaeresis	\(ao

Hex.	Dec.	Okt.	Graphic	Meaning	Input
E6	346	230	æ	Lowercase ae ligature	\(ae
E7	347	231	ç	Lowercasec cedilla	\(cc
E8	350	232	è	Lowercase e grave accent	\(e`
E9	351	233	é	Lowercase e acute accent	\(e'
EA	352	234	ê	Lowercase e circumflex	\(e^
EB	353	235	ë	Lowercase e umlaut or diaeresis	\(e:
EC	354	236	ì	Lowercase i grave accent	\(i`
ED	355	237	í	Lowercase i acute accent	\(i'
EE	356	238	î	Lowercase i circumflex	\(i^
EF	357	239	ï	Lowercase i umlaut or diaeresis	\(i:
F0	360	240	ð	Lowercase Icelandic ETH	**missing**
F1	361	241	ñ	Lowercase n tilde	\(n~
F2	362	242	ò	Lowercase o grave accent	\(o`
F3	363	243	ó	Lowercase o acute accent	\(o'
F4	364	244	ô	Lowercase o circumflex	\(o^
F5	365	245	õ	Lowercase o tilde	\(o~
F6	366	246	ö	Lowercase o umlaut or diaeresis	\(^o
F7	367	247	÷	Division sign	\(di
F8	370	248	ø	Lowercase o crossbar	\(o/
F9	371	249	ù	Lowercase u grave accent	\(u`
FA	372	250	ú	Lowercase u acute accent	\(u'
FB	373	251	û	Lowercase u circumflex	\(u^
FC	374	252	ü	Lowercase u umlaut or diaeresis	\(^u
FD	375	253	ý	Lowercase y acute accent	**missing**
FE	376	254	þ	Lowercase Icelandic Thorn	**missing**
FF	377	255	ÿ	Lowercase y umlaut or diaeresis	\(y`

The following special characters are only available with PostScript format. They do not generate `nroff` output.

Input	troff-Output	Name/Meaning
\(~~	≈	approxequal
\(!b	⊄	notsubset
\(!m	∉	notelement
\(ab	↔	arrowboth
\(af	ℵ	aleph
\(ah	—	arrowhorizex
\(an	∠	angle
\(av	|	arrowvertex
\(b<	⟨	angleleft
\(b>	⟩	angleright
\(c.	·	dotmath
\(cg	≅	congruent
\(el	…	ellipsis
\(fa	∀	universal
\(l&	∧	logicaland
\(lx	⎢	bracketleftex
\(lz	◊	lozenge
\(l|	∨	logicalor
\(pm	‰	perthousand
\(pp	⊥	perpendicular
\(rx	⎥	bracketrightex
\(st	∋	suchthat
\(te	∃	existential
\(tf	∴	therefor
\(ws	℘	weierstrass
\(CR	↵	carriagereturn
\(lf	ℑ	lfraktur
\(LB	⎩	parentleftbt
\(LT	⎧	parentlefttop
\(LX	⎪	braceex

Input	troff-Output	Name/Meaning
\(O+	⊕	circleplus
\(Ox	⊗	circlemultiply
\(Rf	ℜ	Rfraktur
\(RB	⎫	parentrightbt
\(RT	⎪	parentrighttop
\(RX	⎪	parentrightex
\(TM	™	trademarkH
\(tm	™	trademarkR
\(Sl	ℓ	Script-l
\(Sv	Š	Scaron
\(sv	š	scaron
\(Zv	Ž	Zcaron
\(zv	ž	zcaron
\(fr	/	fraction
\(qs	'	quotesingle
\(``	"	quotedblleft
\(q<	‹	guilsinglleft
\(q>	›	guilsinglright
\(en	–	endash
\(qb	‚	quotesinglbase
\(,,	„	quotedblbase
\("	„	quotedblright
\(~a	~	tilde
\(-a	¯	macron
\(oa	°	ring
\(Ua	˘	breve
\(.a	˙	dotaccent
\(:a	¨	dieresis
\("a	˝	hungarumlaut
\(va	ˇ	caron
\(Ca	¸	ogonek
\(L/	Ł	Lslash
\(l/	ł	lslash
\(i.	ı	dotlessi
\(ob	°	base ring

Input	troff-Output	Name/Meaning
\(OE	Œ	OE
\(oe	œ	oe
\(fn	ƒ	florin
\(t1	ϑ	theta1
\(p1	φ	phi1
\(o1	ϖ	omega1
\(U1	ϒ	Upsilon1
\(l	{	braceleft
\(r	}	braceright
\(cl	♣	club
\(dm	♦	diamond
\(ht	♥	heart
\(sd	♠	spade
\(se	″	second
\(pr	∏	product
\(su	∑	summation
\(-b	⇔	arrowdblboth
\(-l	⇐	arrowdblleft
\(-u	⇑	arrowdblup
\(-r	⇒	arrowdblright
\(-d	⇓	arrowdbldown
\(lt	⌠	integraltp
\(lx	⎮	integralex
\(lb	⌡	integralbt

NOTES

The representation of the ', ´, and ` characters can be somewhat strange, depending on whether they are

1)
formatted with `nroff` or `troff`;

2)
or are edited directly as they are either masked (with \) or as special characters (\(*cc*).

The various possibilities are outlined below. □

Input	troff-Output	nroff-Output	Name/Meaning
\'	´	'	Accent aigu
\`	`	`	Accent grave
\(aa	´	'	Accent aigu
\(ga	`	`	Accent grave
\(fm	`	'	foot mark
´	´	´	Accent aigu (=oktal 264)
`	'	`	Accent grave (=oktal 140)
'	'	'	Apostroph (=oktal 047)
\(qs	'		quotesingle

* Warning:
  `nroff` outputs \' and \(aa as a single quote and not as '(acute accent).

SEE ALSO

`meta2fun`(1M), `nroff`(1M), `troff`(1M), `ascii`(5), `meta-ascii`(5), `nterm`(5).

System Administrator's Reference Manual

SIcons(5)
RM400 only

NAME

`SIcons -` package for configuring the console

DESCRIPTION

`pkgadd SIcons` is used to configure and install a multiuser (DPTG) console while `pkgrm SIcons` deinstalls it again.

The `SIcons` package can be used when connecting a 9766, BA80, BA47 console or a PC with corresponding emulation. The defaults for the console parameters are as follows:

`8` Bit

`1` Stopbit

`no` Parity

`19200`
Baud

`SIcons` is an optional package and cannot be updated automatically or installed over. The system must be restarted after each installation and deinstallation or deinstallation followed directly by a reinstallation.

It is also possible to install the `SIcons` package from the console using `cdinst`. However, this type of installation only stores the package (`pkgtrans`) under `/var/spool/pkg` and allows all products from part4 of the installation CD to be installed simultaneously.

For the complete `SIcons` installation, the `pkgadd` command which generally installs the packages stored under `/var/spool/pkg` must be called explicitly.

A complete `SIcons` installation must not be carried out on the console terminal. The DPTG protocol must be activated, i.e. after the `SIdptg2` package has been installed `idbuild` must be run and the system must be started.

How the DPTG console should be configured (e.g. number of windows, printer ports, country-specific keyboard settings, etc.) is queried during the installation.

Directly after the `SIcons` package has been installed successfully, an unloaded terminal can be loaded (see under manual intervention). The correctness of the firmware selected during the installation can therefore be checked. Otherwise, a DPTG console in VT-PROM mode (i.e. not loaded) is loaded on a system with `SIcons` installed when the system is started. Manual intervention [`change to loader` in the setup menu or switching on/off the terminal (switching the keyboard off and on again is not sufficient)] is required in this phase of the booting process.

Note:
The firmware is always selected using the installation parameters. The setting for selecting the firmware in the setup menu on the workstation has no effect.

A change in the loading program (e.g. after changing the keyboard) can also be made by issuing the `pkgadd` command for `SIcons` again, if `SIcons` has already been installed in full.

Deinstallation can only take place successfully when the console ports are not active. Interactive support is provided for aborting the corresponding processes and continuing the `pkgrm`.

Note:
If the specifications are not followed, inconsistencies may arise in the system status.

SEE ALSO

`idbuild`(1M), `pkgadd`(1M), `pkgrm`(1M), `pkgtrans`(1M), `sac`(1M), `sacadm`(1M).

System Administrator's Reference Manual

specialchar(5)

NAME

`specialchar -` shell metacharacters

ARGUMENT AND COMMAND SEPARATORS

Metacharacter	Meaning

`Blank`
`Newline`
`Tab`
	Argument separators, depending on the value of the `IFS` variable

`Newline`	End of command
`|`	Pipe symbol
`;`	End of command
`&`	End of command; a command terminated with this symbol is started in the background.
`||`	ORIF; the next command is executed only if the preceding command returns a non-zero exit status.
`&&`	ANDIF; the next command is executed only if the preceding command returns an exit status of zero.

COMMAND GROUPING

Metacharacter	Meaning
(*command_list*)	Execute *command_list* in a subshell
{ *command_list*; }	Collect the output of all commands from *command_list*

COMMAND SUBSTITUTION

Metacharacter	Meaning
`` `command` ``	Replace *command* with its output

FILENAME GENERATION

Metacharacter	Meaning
*	As a pattern on its own: replaced by the list of all file names in the current directory that do not begin with a period ".". □ As part of a pattern: replaced by zero, one, or more characters to match the file names matching the pattern.
?	As a pattern on its own: replaced by the list of all file names in the current directory that consist of exactly one character, except for a period. □ As part of a pattern: replaced by exactly one character to match the file names matching the pattern.
[*s*]	Replaced by exactly one character from the character string *s* to match the file names matching the pattern.
[*c1*-*c2*]	Replaced by exactly one character from the range between *c1* and *c2* (*c1* and *c2* inclusive) to match the file names matching the pattern. *c1* and *c2* must be

	ordinary characters. The characters which collate in the range *c1-c2* are determined by the ASCII collating sequence. □ Expressions of type [*s*] and [*c1-c2*] can be combined: [*s1c1-c2s2*].
[!*s*]	Replaced by exactly one character excluding those contained in the character string *s* to match the file names matching the pattern.
[!*c1-c2*]	Replaced by exactly one character excluding those in the range *c1* through *c2* inclusive to match the file names matching the pattern (see [*c1-c2*]). □ Expressions of type [!*s*] and [!*c1-c2*] can be combined: [!*s1c1-c2s2*].

[[:*class*:]]	(`ksh` only) Replaced by exactly one character from the character class *class*. *class* can be: □ `alpha`□□any letter□ `upper`□□any uppercase letter□ `lower` any lowercase letter□ `digit` any decimal digit (0 through 9)□ `xdigit` .i1 +5m any hexadecimal digit (0 through 9, a through f, and A through F)□ `alnum`□□any alphanumeric character (letters and digits)□ `space`□□any character producing white space in displayed text (e.g. blanks and tabs)□ `blank`□□blanks or tabs□ `punct`□□any separator□ `print`□□any printable character (including the characters in `space`)□ `graph`□□any printable character with a visible representation (excluding the characters in `space`)□ `cntrl`□□any control character
[[=*c*=]]	(`ksh` only) Replaced by exactly one character or a collating element that has the same relative order as *c*. *c* must not be an equals sign = or a right square bracket].
[[.*cc*.]]	(`ksh` only) Replaced by precisely one collating element. Multi-character collating elements must be represented in this form to distinguish them from ordinary characters. *cc* has to be defined as a valid collating element in the internationalized environment.

REDIRECTION OF STANDARD OUTPUT

Metacharacter	Meaning
>*file*	Redirect standard output to *file*, deleting original contents
>>*file*	Redirect standard output to *file*, retaining original contents
>&*digit*	Redirect standard output to the file associated with the file descriptor *digit*
>&-	Close standard output

REDIRECTION OF STANDARD INPUT

Metacharacter	Meaning
<*file*	Redirect standard input to *file*
<<*argument*	Start a "here" document
<<-*argument*	Start a "here" document, stripping leading tabs
<&*digit*	Redirect standard input to the file associated with the file descriptor *digit*

`<&-`	Close standard input

SHELL VARIABLES AND PARAMETERS

Metacharacter	Meaning
`name=`*value*	Assign a value to the variable *name*
`$`*name*	Value of the variable *name*; keyword parameter
`${`*name*`}`	Same as `$`*name*; the braces delimit the variable name from following digits or letters.
`${`*name*`-`*default_value*`}`	Replace with *default_value* if *name* is undefined
`${`*name*`=`*default_value*`}`	Assign *default_value* if *name* is undefined
`${`*name*`?`*default_value*`}`	Shell terminates execution with error message *parameter* `:` *default_value* if *name* is undefined
`${`*name*`+`*default_value*`}`	Replace with the null string if *name* is undefined. Replace with *default_value* if *name* is defined.
`${`*name*`:-`*default_value*`}`	Replace with *default_value* if *name* is undefined or its value is the null string
`${`*name*`:=`*default_value*`}`	Assign *default_value* if *name* is undefined or its value is the null string
`${`*name*`:?`*default_value*`}`	Shell terminates execution with error message *parameter* `:` *default_value* if *name* is undefined or its value is the null string
`${`*name*`:+`*default_value*`}`	Replace with the null string if *name* is undefined or its value is the null string. Replace with *default_value* if *name* is defined and its value is not the null string.
`$0`	0th command line argument, i.e. the command name, the shell script, or the current shell
`$1, $2, ... , $9`	Positional parameters
`$*`	All command line arguments
`"$*"`	All command line arguments as a single argument
`$@`	All command line arguments
`"$@"`	All command line arguments as separate arguments
`$#`	Number of command line arguments, i.e excluding `$0`
`$$`	Process ID (PID) of the current shell
`$!`	PID of the last command started in the background
`$?`	Exit status of the last executed command that was not started in the background
`$-`	All options set in the current shell

SHELL FUNCTIONS

Metacharacter	Meaning
`name() {` *command_list*`;}`	Shell function; when *name* is invoked, the commands from

	command_list are executed

QUOTING METACHARACTERS

Metacharacter	Meaning
\	Escapes the next metacharacter
'...'	Escapes all metacharacters; treat contents as a single argument
"..."	Does not escape the metacharacters $, `...` and \; treat contents as a single argument

MISCELLANEOUS

Metacharacter	Meaning
#	Comment character in shell scripts
;;	Command list terminator within case constructs

SEE ALSO

csh(1), ksh(1), sh(1).

System Administrator's Reference Manual

statis(5)

NAME

`statis` - kernel statistics facilities

SYNOPSIS

```
#include <sys/types.h>
#include <sys/param.h>
#include <sys/kernel.h>
#include <sys/var.h>
#include <sys/tuneable.h>
#include <sys/sysinfo.h>
#include <sys/immu.h>
```

DESCRIPTION

The system may contain many different statistics facilities. Some are specific to particular applications and some will exist on every system. Following is a list of the common facilities. A complete list of the facilities present on a system may be obtained using `/sbin/statis -t`.

Facility	Type	Description
ncpu	unsigned	number of cpus
nproc	unsigned	current size of the process table
physmem	unsigned	number of PAGESIZE pages in physical memory
boottime	time_t	time at which system was booted in seconds
avenrun	long[3]	5, 10, and 15 minute load averages
var	struct var	system configuration information
tune	struct tune	system tunable information
sysinfo	struct sysinfo	process, IO, and virtual memory statistics
minfo	struct minfo	virtual memory statistics
vminfo	struct vminfo	virtual memory statistics
syserr	struct syserr	table overflow counts
kmeminfo	struct kmeminfo	kernel heap statistics
vm cnt	struct vmmeter	virtual memory short-term counts
vm rate	struct vmmeter	virtual memory 5 second average rates
vm sum	struct vmmeter	virtual memory total counts
vm total	struct vmtotal	virtual memory totals
maxvdunit	unsigned short	maximum number of virtual disks

SEE ALSO

`statis(3X)`, `statis(8)`.

System Administrator's Reference Manual

taclan(5)

NAME

`taclan -` TACLAN initialization and configuration

DESCRIPTION

Devices on network stations, such as terminal servers or printer servers, are configured via SNMPv2. There is some vendor-specific information to be noted here:

Device type ANNEX

These devices are not recognized automatically, and must therefore first be initialized:

- The current contents of EEPROMs are first deleted using the monitor command "erase". The most important settings for network operation are then entered using the monitor command "addr".
- If the device is to be loaded via another system, the `SIadm863` package (at least Version 2.0) must be available on that system. The line

  ```
  snmp community public
  ```

  must be added to the `gateways` file in the `/opt/adm863` directory. This means that this network station responds to all SNMP requests which are received with the key "`public`". Full access protection can only be guaranteed if the network stations can work with the v2 Version of SNMP. No security functions can be activated until this condition is met.
- This device is then loaded using the "boot" command.
- The connected terminal is connected to any (free) port and the `su` command is entered via the CLI. The keyword is the Internet address of the device itself.
- The following is then set using the "admin" command: `set annex allow_snmp_sets Y`
- The next command to be entered (`su`) terminates this action.
- Once the device has been loaded using "boot", it can be configured from any Reliant UNIX system.

Only some of the functions of an ANNEX that can be set are offered for configuration by default. It is, however, possible to specify expansions (e.g. switching on the "routing"). In this case, an attribute set is created for the network station function to be set [see `mibtool(1M)`]. This attribute set is written to the `/var/sadm/term/tsvrtypes/annex8/add-on.def` file. The name of the attribute set can be specified in the configuration, and the corresponding settings are then made at the end of the configuration process. The `config.def` file, which is also available in this directory, must not be modified. The `misc.def` file can be adapted to specific situations.

When using CLI configurations, access must always be via `telnet -t` *internet_address port_>7000*.

"Dedicated" configurations are supported, but not in the context of the RTY procedure.

FILES

```
/var/sadm/term/tsvrtypes/annex8/config.def
/var/sadm/term/tsvrtypes/annex8/misc.def
/var/sadm/term/tsvrtypes/annex8/add-on.def
```

SEE ALSO

`mibtool(1M)`, `tsvrset(1M)`, `tsvrget(1M)`.

System Administrator's Reference Manual

troff(5)

NAME

`troff` - description of output language

DESCRIPTION

Output from `troff` is pure ASCII written in the following language:

`s`*n* Use point size *n*.

`f`*n* Use the font mounted at position *n*.

`c`*x* Print ASCII character *x* at the current position.

`C`*xy* Print special character *xy* at the current position.

`H`*n* Absolute horizontal move to position *n*.

`h`*n* Relative horizontal move of *n* units.

`V`*n* Absolute vertical move to position *n* (down is positive).

`v`*n* Relative vertical move of *n* units.

nnx Move *n*n units horizontally then print ASCII character *x*.

`n`*b a* End of a line.

`p`*n* Begin page *n*.

`#` ...
Comment – to the end of the line.

`Dl` *dx dy*
Line from current point (*x*, *y*) to point (*x*+*dx*, *y*+*dy*).

`Dc` *d* Circle of diameter *d* with left side at the current point.

`De` *dx dy*
Ellipse with axes *dx* and *dy* and left side at the current point.

`Da` *dx1 dy1 dx2 dy2*
Counterclockwise arc from current point (*x*, *y*) to (*x*+*dx1*+*dx2*, *y*+*dy1*+*dy2*). Center of the circle is (*x*+*dx1*, *y*+*dy1*).

`D~` *dx1 dy1 dx2 dy2 ...*
B-spline from current point (*x*, *y*) to (*x*+*dx1*, *y*+*dy1*), then to (*x*+*dx1*+*dx2*, *y*+*dy1*+*dy2*).

`x init`
Initialize the typesetter.

`x T` *device*
The *n*ame of the typesetter is *device*.

`x res` *n h v*
The resolution of the typesetting device in increments per inch is *n*. Motion in the horizontal direction can take place in units of *h* basic increments. Motion in the vertical direction can take place in units of *v* basic increments.

`x stop`
Stop – done forever.

`x trailer`
Generate a trailer.

`x font` *n name*
Mount font *name* at position *n*.

`x Height` *n*
Set character height to *n* points.

`x Slant` *n*
Set character slant to *n* degrees.

SEE ALSO

`troff`(1M) .

REFERENCE

B. W. Kernighan, *A Typesetter-Independent Troff*

System Administrator's Reference Manual

utsdbg(5)
RM600 only

NAME

`utsdbg -` life and death of the debug kernel

DESCRIPTION

An overview of the generation, activation, and deletion of the debug kernel is given in this manpage and explained using an example. Reference is made there to the relevant commands and their manual pages.

INSTALLING

You want to install the debug kernel environment.

The debug kernel environment cannot yet be installed on the machine. Use the `pkginfo utsdbg` command to check whether one of the debug kernel packages, namely `utsdbg`, is already installed. If it is, you need to delete the old environment using

```
del_dbg_kernel -a
```

before installing the new one. Refer here to the manpage for `del_dbg_kernel(8)` .

Now check whether there is enough space in `/opt/.confutsdbg`. You will need approx. 55,000 Kbytes free. If there is not enough space, you need to allow the symbolic link `/.utsdbg/etc/conf` to point to another file system where there is enough space.

Insert the correct operating system CD into the CD-ROM drive and mount the CD.

Example:

```
mount -F hs /dev/ios0/sdisk006s0 /cdrom
```

Read the manual page for `inst_dbg_kernel(8)` and now set up the debug kernel environment using:

```
inst_dbg_kernel
```

If `inst_dbg_kernel` issues a warning that single correctional releases (SCR) are installed on your machine, check whether these releases contain kernel components. If they do, you need to get the correct debug kernel components for these SCRs and install them. Otherwise, your debug kernel will not contain the error recoveries from the SCRs that affect the kernel.

LINKING

The debug kernel should now be linked. You may need to note the old value of the `ROOT` variable and set the `ROOT` variable as follows:

```
ROOT=/.utsdbg ; export ROOT
```

Now call `idbuild` in the debug kernel environment. The `idbuild` command evaluates the value of the `ROOT` variable for further internal calls:

```
$ROOT/etc/conf/bin/idbuild -L
```

Information on the `idbuild` command can be found in the `idbuild(1M)` manual page.

After the `$ROOT/etc/conf/bin/idbuild -L` command, reset the `ROOT` variable to its old value.

ACTIVATING

You now want to activate the debug kernel.

Read the `setup_dbg_kernel(8)` manual page and invoke the following command:

```
/.utsdbg/etc/conf/bin/setup_dbg_kernel -a
```

The debug kernel is activated with the next reboot (unless a `/etc/conf/bin/idbuild -M` command is executed successfully; see `setup_dbg_kernel(8)` manual page).

DEACTIVATING

Read the information contained in the setup_dbg_kernel(8) manual page.

You have already booted the debug kernel and now want to boot a "normal" kernel again.

Delete the ROOT variable with unset ROOT, since idbuild evaluates the value of the ROOT variable. Link a "normal" kernel with:

```
/etc/conf/bin/idbuild -M
```

and then perform a reboot.

DELETING

The debug kernel environment should be deleted on your machine.

Read the del_dbg_kernel(8) manual page here. It is called as follows:

```
del_dbg_kernel -a
```

If there is an update to another operating system version, the debug kernel environment is no longer valid, and is therefore deleted during the update. The environment must be set up again after the update with inst_dbg_kernel.

DIAGNOSTICS

The individual commands issue error messages, which can be referenced in the relevant manual pages.

SEE ALSO

idbuild(1M), del_dbg_kernel(8), inst_dbg_kernel(8), setup_dbg_kernel(8).

System Administrator's Reference Manual

XXexist(5)
RM400 only

NAME

`XXexist -` determine if a device exists

SYNOPSIS

```
#include <sys/infconfig.h>
int XXexist(intr_param_t *);
```

DESCRIPTION

The `XXexist` routine is the first function called for a given driver. Its function is to determine if the driver has any hardware to control. The information given in the `intr_param_t` structure can be used to differentiate between various pieces of hardware.

The `intr_param_t` structure consists of the following definitions:

```
typedef struct  intr_params {
        int     bus_type;
        int     unit;
        struct ioaddress_info   ioports[1];
        int     __spare[12];
} intr_param_t;
```

The members of this structure are:

`bus_type`
: This specifies that the hardware is either on the EISA or ISA bus.

`unit`
: This value is taken from the `unit` field of the `sdevice` file. It can be used to determine which device is being referenced.

`ioports`
: This structure is not currently used, but may in a later revision of the Reliant UNIX system allow the driver to determine which I/O ports on the ISA bus should be used.

The `XXexist` routine can return one of two values. If the hardware does not exist, then the routine should return `PROBE_FAIL`. If the hardware exists or if the routine cannot determine if the hardware exists, then the routine should return `PROBE_OK`. `PROBE_OK` and `PROBE_FAIL` are both defined in `intconfig.h`.

The `XXexist` routine is called before any interrupts are enabled, so interrupts cannot be used. The `XXexist` routine cannot allocate memory.

FILES

`sys/intconfig.h`

XXsetint(5)
RM400 only

NAME

`XXsetint -` inform a driver of a given interrupt

SYNOPSIS

```
#include <sys/infconfig.h>
void XXsetint(int *, intr_param_t *);
```

DESCRIPTION

The `XXsetint` routine is called after the `XXexist` routine, but before any other routine in the driver. The `XXsetint` routine is called when the Reliant UNIX system has determined which interrupt number a given piece of hardware should use. The `intr_param_t` parameter can be used to differentiate between various calls to this routine. The first parameter to this routine is a pointer to a list of interrupt numbers which the driver must use for this device. If the `XXsetint` routine is not called for a given device, the driver should not be initialized since, no interrupt could be allocated to the device. The `XXsetint` routine will be called for each line in the drivers `idevice` file.

The `intr_param_t` structure consists of the following definitions:

```
typedef struct  intr_params {
        int     bus_type;
        int     unit;
        struct ioaddress_info   ioports[1];
        int     __spare[12];
} intr_param_t;
```

The members of this structure are:

`bus_type`
: This specifies that the hardware is either on the EISA or ISA bus.

`unit` This value is taken from the `unit` field of the `sdevice` file. It can be used to determine which device is being referenced.

`ioports`
: This structure is not currently used, but may in a later revision of the Reliant UNIX system allow the driver to determine which I/O ports on the ISA bus should be used.

The `XXsetint` routine is called before any interrupts are enabled, so interrupts cannot be used. The `XXsetint` routine cannot allocate memory.

FILES

`sys/intconfig.h`

Special Files(7)

Introduction

intro_sarm7(7)

NAME

`intro_sarm7 -` introduction to special files

DESCRIPTION

This section describes various special files that refer to specific hardware peripherals, and Reliant UNIX

system device drivers. STREAMS [see intro_prm2(2)] software drivers, modules and the STREAMS-generic set of ioctl(2) system calls are also described.

For hardware related files, the names of the entries are generally derived from names for the hardware, as opposed to the names of the special files themselves. Characteristics of both the hardware device and the corresponding Reliant UNIX system device driver are discussed where applicable.

abiaio (7) - filesystem(7)

System Administrator's Reference Manual

abiaio(7)

NAME

abiaio:
aio_read, aio_write, lio_listio, aio_fsync, aio_error, aio_return, aio_cancel, aio_suspend, aio_hold - MIPS-ABI-aio library interfaces

SYNOPSIS

```
#include <sys/types.h>
#include <errno.h>
#include <limits.h>
#include <unistd.h>
#include <signal.h>
#include <time.h>
#include <sys/aio.h>
int aio_read(aiocb_t *aiocbp);
int aio_write(aiocb_t *aiocbp);
int lio_listio(int mode, const aiocb_t *list[], int nent,
                sigevent_t *sig);
int aio_fsync(aiocb_t *aiocbp);
int aio_error(aiocb_t *aiocbp);
int aio_return(aiocb_t *aiocbp);
int aio_cancel(aiocb_t *aiocbp);
int aio_suspend(const aiocb_t *list[], int nent,
                const struct timespec *timeout);
int aio_hold(int should_hold);
int is_aio(int fildes);
```

DESCRIPTION

aiocbp job structure

aiocbp points to an asynchronous I/O control block with the following structure:

```
typedef struct aiocb {
        int     aio_fildes;       /* file descriptor */
        volatile
        void    *aio_buf;         /* data buffer location */
        size_t  aio_nbytes;       /* number of data bytes */
        off_t   aio_offset;       /* file offset */
        int     aio_reqprio;      /* request priority */
        struct sigevent aio_sigevent;   /* notification method */
        int     aio_lio_opcode;   /* listio operation */
        ulong   aio_reserved[7];  /* reserved for internal use */
        ulong   pad[6];           /* reserved for expansion */
```

```
} aiocb_t;
/* aio_offset interpretation flag */
#define aio_blkseek_flag    aio_reserved[4]
```

The `aio_fildes` element is the file descriptor for which the asynchronous operation is to be executed. If `aio_fildes` is allocated to a direct access device, the requested operation takes place at the absolute position in the file specified by `aio_offset`, as if `lseek` had been called immediately before the requested operation with an offset argument equal to `aio_offset` and the `whence` argument equal to `SEEK_SET`. See also the section about blockseek job enhancement, further below. If the system call for the operation was successful and it was queued, the value of the file offset is undefined afterwards.

The `aio_nbytes` element specifies the number of bytes for reading or writing.

The `aio_buf` element refers to the buffer used during I/O transfer.

The `aio_reqprio` element is currently ignored since `_POSIX_PRIORITIZED_IO` and `_POSIX_PRIORITY_SCHEDULING` are not supported.

The `aio_sigevent` element defines how the process making the call is notified when the I/O is terminated. The detailed structure is shown below.

The `aio_lio_opcode` field is only used in the `lio_listio()` call. The `lio_listio()` call permits the transfer of a number of asynchronous I/O operations using a single function call. The function accepts an array of pointers to the `aiocb` structures as an argument. Each `aiocb` structure describes an I/O job. The value in the `aio_lio_opcode` field specifies whether a read or write operation is to be performed.

The address of the `aiocb` structure is used as a reference for calling the error status and return status of the asynchronous operation while it is being executed.

The `aiocb` structure and the data buffers allocated to the I/O operation are only used by the operating system for asynchronous I/O operations while the error status of the asynchronous operation is equal to `EINPROGRESS`. Applications may not change the `aiocb` structure while this structure is being used by the system for asynchronous I/O operations.

The return value of the asynchronous operation is the number of bytes transferred by the I/O operation. If an error status is returned, i.e. if an error occurred during the operation, the return value is that which would have been returned by the `read()`, `write()`, or `sync()` system call. The return value is only defined if the error status is not equal to `EINPROGRESS`.

The following information must be noted in relation to non MIPS ABI-compliant blockseek job enhancement.

The job specification was extended to enable raw disk files in excess of 2 Gbytes to be addressed.

The `aio_reserved[4]` parameter is used as an additional job specification in the `aiocb` job field. It can be addressed via the name `aio_blkseek_flag` (see also `sys/aio.h`).

If the `AIO_BLK_OFFSET` value is set in `aio_blkseek_flag`, the value set in `aio_offset` is interpreted as a logical disk sector size unit (512 Bytes).

If a value other than `AIO_BLK_OFFSET` is set in `aio_blkseek_flag`, the value in `aio_offset` is assumed as before as the byte offset.

With the release of this additional functionality, the `aio_blkseek_flag` job parameter must be set in the job definition of an AIO job for `aio_read()`, `aio_write()` and `lio_listio()`.

Signal-specific structures

The following is a detailed description of the `sigevent` structure:

```
typedef struct sigevent {
   int  sigev_notify;                /* notification type */
   notifyinfo_t  sigev_notifyinfo;   /* notification info */
   sigval_t  sigev_value;            /* signal value */
   ulong  sigev_reserved[13];        /* reserved for internal use */
   ulong  sigev_pad[6];              /* reserved for future use use */
} sigevent_t;
typedef union sigval {
   int   sival_int;                 /* integer notify value */
```

```
   void  *sival_ptr;                /* pointer notify value */
} sigval_t;
typedef union notifyinfo {
   int   nisigno;                    /* signal number */
   void  (*nifunc)(sigval_t );       /* callback data */
} notifyinfo_t;
#define sigev_func              sigev_notifyinfo.nifunc
```

The `sigev_notify` field specifies the notification type when the I/O operation is terminated. If the value is `SIGEV_NONE`, no signal is sent when the I/O operation is terminated. If the value is `SIGEV_CALLBACK`, the `SIGEMT` signal is output. This signal is processed by a routine configured by the `abiaio` library functions. This means that the application program may not use this signal when using the asynchronous I/O interface.

`sigev_notifyinfo` is a union that contains the notification information. The `nisigno` field is not used currently. The `nifunc` field contains the address of the function that is to be called by the library when the I/O operation is terminated.

`sigev_sigval` is a union that contains the parameters for the callback function. Applications use the `sigval_int` field if the argument is an integer. If it is a pointer, they use the `sigval_ptr` field.

Timeout-specific structure

The following structure is used by applications that call `aio_suspend()` to forward the value for the timer:

```
typedef struct timespec {
      time_t  tv_sec;          /* seconds */
      time_t  tv_nsec;         /* nano seconds */
} timespec_t;
```

The implementation-specific internal maximum value for the timer is approximately 2100 seconds.

Description of functions

The process can start an asynchronous read operation using the `aio_read()` function. The single argument `aiocbp` points to a control block in the address space of the calling process in which the I/O task is defined. If the asynchronous I/O request is terminated successfully, `aio_nbytes` are transferred from the file with the `aio_fildes` file descriptor to the buffer referred to by `aio_buf`.

The function call returns if the read task was initialized or included in the device queue. The requested operation takes place at the absolute position in the file specified by `aio_offset`, as if `lseek()` had been called immediately before the operation with an offset argument equal to `aio_offset` and the `whence` argument equal to `SEEK_SET`. See also the section about blockseek job enhancement, further above. If the system call for the operation was successful and it was queued, the value of the file offset is undefined afterwards.

If an error occurred and the job is not queued, the function returns –1 and `errno` is set to the appropriate value.

If there is no error detected in the function call, `aio_read()` returns the value 0. The calling process may not change the control block while the I/O task is being processed in the operating system since the operating system submits status information to the control block. The process must call `aio_error()` and `aio_return()` to query the error status or return status of the asynchronous operation.

The calling process can use the callback function to be informed immediately when the I/O operation is terminated. To activate this function, `sigev_notify` in the `aio_sigevent` field must have the value `SIGEV_CALLBACK`, `sigev_func` must contain the address of the callback function, and `sigev_value` must contain the value that is transferred as a parameter when the callback function is called.

The operating system sends a `SIGEMT` signal to a process when the task is terminated. The signal is processed by a library function and means that the callback function is called.

The `aio_lio_opcode` field is ignored by `aio_read()`. Modifying `aio_buf` or `abiocbp` while the task is being processed in the operating system can result in undefined events. Simultaneous asynchronous operations that use the same `aiocbp` job field create undefined events. `aio_read()` only supports access to disk files in RAW mode.

_POSIX_SYNCHRONIZED_IO is not defined and is not supported by the operating system.

_POSIX_PRIORITIZED_IO is not supported.

Using the aio_write() function, the calling process can write aio_nbytes bytes from the buffer pointed to by aio_buf to the file with the file descriptor aio_fildes.

The description for aio_read() is the same.

aio_write() only supports access to disk files in RAW mode. The O_APPEND function is not supported.

Using the lio_listio() function, the calling process can initiate a list of I/O requests with a single function call.

The value LIO_WAIT or LIO_NOWAIT must be specified for mode. The value defines whether the function returns after termination of the I/O operations or after inclusion of the operations in the queue. If the mode argument has the value LIO_WAIT, the function waits until all I/O operations are terminated, and the sig argument is ignored.

If the mode argument has the value LIO_NOWAIT, the function returns immediately. The calling process can also use the callback functionality. To activate this function, the sigev_notify of the sig argument must have the value SIGEV_CALLBACK, sigev_func must contain the address of the callback function, and sigev_value must contain the parameter value of the callback function. The operating system sends the SIGEMT signal when the I/O operations are terminated. The calling process may not use this signal itself. If sig is NULL or the value of sigev_notifyinfo is NULL, no signal is sent. The processing sequence of the I/O jobs is arbitrary and need not correspond to the sequence in the list.

The list argument is an array of pointers to aiocb structures. The array contains nent elements.

The aio_lio_opcode field of each aiocb structure specifies the operation to be executed. The values LIO_READ, LIO_WRITE, and LIO_NOP are permitted. These trigger a corresponding asynchronous I/O operation. If the field contains the value LIO_NOP, the list entry is ignored.

The calling process can also use the callback function for one or more jobs in the list. The mode argument must be LIO_NOWAIT. The aio_sigevent of the I/O control block must then be defined as described under aio_read().

You can find information on the meaning of the different fields of the I/O control block as well as the operation itself in the description of aio_read() and aio_write().

The aio_fsync() function forces the synchronized I/O termination status of all I/O operations for the file specified by the aio_fildes file descriptor. The synchronized I/O termination status is achieved if the function is returned without an error.

In the current implementation, the synchronized I/O termination status is achieved if the I/O operations are terminated for all affected jobs.

The calling process transfers the address to an I/O control block in aiocbp. In the I/O control block, aio_fildes must be set defined.

The op argument is either O_DSYNC or O_SYNC. Since aio is only supported on devices in RAW mode, no distinction is made between the two values.

If aio_fsync() returns without an error and is terminated, all I/O requests occurring up to this point for the specified file are processed until they have achieved the synchronized I/O termination status. Jobs initiated after aio_fsync() is called are not taken into account.

If the aio_fsync() call returns an error, or an error status is returned for aio_fsync(), no information on the correct completion of the relevant I/O requests is available.

The aio_error() function transfers the address to an I/O job. The corresponding error status is supplied as the return value. The error status for an asynchronous operation is errno, as was set by the corresponding read, write, or fsync operation. If the operation is not yet terminated, the error status is equal to EINPROGRESS.

The aio_return() function returns the status allocated to the aiocb structure to which the aiocbp argument refers. The return status for an asynchronous I/O operation is the value that was returned by the corresponding read, write, or fsync operation. If the error status for the operation is equal to

EINPROGRESS, the return status for the operation is undefined.

The aio_cancel() function attempts to cancel one or more asynchronous I/O jobs currently outstanding for the fildes file descriptor. If the aiocbp argument refers to an asynchronous I/O control block for a specific job, it is canceled. If aiocbp is equal to NULL, all outstanding asynchronous I/O requests for fildes that can be canceled are canceled.

The callback function remains active even for canceled jobs.

The allocated error status is set to ECANCELED and the return status is set to −1 for jobs that were canceled successfully. Jobs that could not be canceled successfully do not affect the job completion report.

The function aio_suspend() returns to the calling process if

- at least one of the asynchronous I/O operations to which the list refers is terminated or,
- if a signal interrupts the calling process or,
- if timeout is not equal to NULL and the timeout time has elapsed.

The list argument is an array made up of pointers to asynchronous I/O control blocks. The nent argument specifies the number of elements in the array. Each aiocb structure indicated must be defined in accordance with the operation description for aio_write(), aio_read() or lio_listio(). This array can contain NULL pointers that are ignored. The result is undefined if this array contains pointers to aiocb structures without valid asynchronous I/O operations.

If the time interval specified in the timespec structure that timeout references elapses before one of the I/O operations referred to by list is terminated, aio_suspend() returns an error.

Callback functions calls are controlled using the aio_hold() function.

If the should_hold parameter has the value AIO_HOLD_CALLBACK, callback calls are delayed. If it has the value AIO_RELEASE_CALLBACK, callback calls are resumed. If termination successful, the previous setting is returned. Otherwise the value −1 is returned and errno is set to display the error. aio_hold() can also be called with the AIO_ISHELD_CALLBACK parameter that queries the current status. If the hold status is set, the value 1 is returned, otherwise 0 is returned.

When calling a callback function, the calls of other callback functions for this process are delayed, i.e. only one callback function can be active at any point in time.

NOTES

The contents of this manual page and the description of the functions are based on the two documents mentioned below. The POSIX document contains the functional description. Other functionality details can also be found there.

The MIPS-ABI document contains the corresponding data structures and some restrictions for the POSIX document. This document also describes some additional implementation-specific restrictions.

SEE ALSO

"IEEE STD 1003.1b" POSIX P1003.4/D14.1 May 93, Chapter 6.7: Asynchronous Input and Output

"MIPS ABI Conformance Guide" Version 1.2, Chapter 6: ABI LIBRARY

System Administrator's Reference Manual

accioctl(7)

NAME

accioctl - automatic medium changer device control interface

DESCRIPTION

The interface described here is implemented for Siemens-supported application software. Siemens customers are not encouraged to use this interface. The interface may change in future releases.

This interface supports SCSI medium changer (jukebox) devices. See sjuke(7) for information about jukebox device naming conventions.

The raw (character) interface is the only interface, which is supported for jukebox devices. An attempt to open the block jukebox device will result in an ENXIO error being returned by the open routine. The only system calls supported are open(2), close(2), and ioctl(2). Reading and writing is not possible on a jukebox. An attempt to read or write on a jukebox will result in ENODEV.

Jukebox devices are checked and initialized at boot time or by the reinitdev(8) command.

When they are first opened, jukebox devices are configured according to the mode select parameters in acc/space.c. O_EXCL should be used exclusively to open the device. A request to open a device, which is already exclusively opened, or a request to exclusively open a device, which is already opened, will result in EBUSY. If the device cannot be opened due to configuration or hardware problems, ENXIO or EIO will be returned respectively.

When the jukebox devices are closed, no actions are taken on the jukebox device.

The term element is used to refer to a physical location within a jukebox. A jukebox is viewed as a set af addressable elements, each of which may contain a unit of media or be used as a destination into which a unit of media may be moved. There are four types of elements:

Import/export element:
: This element is used to insert media into or remove media from the jukebox.

Storage element:
: This is one of several slots, where media may be stored.

Medium transport element:
: This is a mechanism, which is used for transporting media.

Data transfer element:
: This is a device which can read from/write to the medium, e.g. an optical disk drive or a tape drive.

Within the ioctl interface, elements are addressed using the following structure:

```
typedef enum { Inlet=0, Slot=1, Holder=2, Drive=3 } adrtyp_t;
#define Any 255
typedef struct {
      adrtyp_t typ;            /* Type of element */
      unsigned char reladr;    /* Relative address of element */
}       adr_t;
```

Import/export elements, storage elements, medium transport elements, and data transfer elements are referenced as Inlet, Slot, Holder, and Drive respectively. In the case of medium transport elements (Holder), Any can be specified as the relative address, which means that the device or the driver will select a suitable medium transport element.

During a move from one element to another, the medium can be rotated by the medium transport element. This allows access to a particular side of the medium. Within the ioctl interface, rotation is referenced by the following enumeration type:

```
typedef enum { SideA=0, SideB=1, Invert=2, Dontchange=3 } side_t;
```

If Invert is specified, the medium will be rotated whereas if Dontchange is specified the medium is not rotated. SideA and SideB are not valid for SCSI jukebox devices.

Information and control operations are provided through `ioctl`. These have the following form:

```
#include <sys/accioctl.h>
void *p;
retval = ioctl(fildes, code, p);
```

The `p` argument is a generic pointer whose type depends on the specific `ioctl` code. Where not specifically mentioned below, its value must be zero. `<sys/accioctl.h>` contains definitions of `ioctl` codes and data structures used by the operations. The following is a list of permitted `ioctl` codes:

`ACC_BARCODE`
: This will cause the jukebox to read the barcode label and it returns the barcode for the specified element. `p` is a pointer to `BarCode` structure.

```
struct BarCode {
      adr_t adr;      /* Element,barcode is requested for */
      char  barcode[MAXBAR];  /* the barcode label */
      }
#define MAXBAR 10
```

`ACC_CELLINIT`
: This will cause the jukebox to update its internal tables. Depending on the type of jukebox and the actions taken on the jukebox, `ACC_CELLINIT` should be called before calling `ACC_CELLSTAT`.

`ACC_CELLSTAT`
: This returns the status of the specified element. `p` is a pointer to a `CellStatus` structure:

```
typedef enum { Empty, Full, TBD, BadState } cstat_t;
struct CellStatus {
 adr_t adr;       /* Element whose status is requested */
 cstat_t status;     /* Return value */
};
```

Possible status codes are `Empty` and `Full`. `BadState` is returned in the event of a hardware error or another error situation. Depending on the jukebox type and the error situation, the reason for a `BadState` may be a missing `ACC_CELLINIT` request.

`ACC_DISPLAY`
: This is a dummy, which always returns 0 for compatibility.

`ACC_DISPLAY_MSG`
: This will cause the jukebox to show a specified message on the display in different modes. `p` is a pointer to a `SelectPara` structure.

```
struct SelectPara {
      int flag;                 /* Message Display Control */
      u_char displ_msg[MAXMSG]; /* Messagetext */
      }
#define MAXMSG 60
```

Values for the parameter `flag` are:

```
#define SCMSEL_MDCFLASH 0x00 /* display msg: flash 8 char display */
#define SCMSEL_MDCSTAY  0x01 /* display msg: steady 8 char display */
#define SCMSEL_MDCSCROL 0x02 /* display msg: scroll display 60 char */
```

The parameter `displ_msg` must contain the message.

`ACC_EXCHANGE`
: This requests the jukebox to exchange media between elements. `p` is a pointer to a `ExDsc` structure:

```
struct ExDsc {
 adr_t  trs;       /* Should have a valid or "Any" holder */
 adr_t  src,dst1,dst2;
 side_t side1,side2;
};
```

The medium from `src` is moved to `dst1` and the medium from `dst1` is moved to `dst2`. `trs` specifies the Medium Transport Element to use. `src` and `dst2` may be identical. The medium of the first move will be rotated according to `side1`, the medium of the second move will be rotated according to `side2`.

`ACC_GETCAPFLAG`

This returns the device capability flag as defined in `acc/space.c`. `p` is a pointer to an integer. The following definitions are from `<sys/ios_conf.h>`:

```
#define ACPF_SETDBD     0x0001  /* Set DBD to 1 */
#define ACPF_ANY        0x0002  /* Autom. transport elem. selection */
#define ACPF_NOINVERT   0x0004  /* No change of medium orientation */
#define ACPF_NOEXCHANGE 0x0008  /* "Exchange Medium" not supported */
#define ACPF_NOLOCK     0x0010  /* "Medialock" not supported */
#define ACPF_NOREZERO   0x0020  /* "Rezero" not supported */
#define ACPF_NOMODESEL  0x0040  /* "Mode Select" not supported */
#define ACPF_DISPLAY    0x0080  /* Front Panel Display available */
#define ACPF_RESERVE    0x0100  /* Do Reserve/Release on open/close */
```

`ACC_GETGEO`

This returns the geometry data of the jukebox. `p` is a pointer to a `geometry` structure:

```
struct geometry {
 unsigned short g_nodd;   /* No. of data transfer elements */
 unsigned short g_nslot;  /* No. of storage elements */
 unsigned short g_nhold;  /* No. of medium transport elements */
 unsigned short g_ninlet; /* No. of import/export elements */
};
```

`ACC_INIT_ELEMENT`

This will cause the jukebox to update its internal table for the specified element. `p` is a pointer to a `InitRange` structure.

```
struct InitRange {
      adr_t adr;      /* Element to initialize */
      int nbl;        /* scan - do not scan barcode label */
      }
```

The values for the parameter `nbl` are:

```
#define  SCINIT_BL    0x00  /* scan the barcode label */
#define  SCINIT_NBL   0x80  /* do not scan the barcode label */
```

`ACC_INLETINOUT`

This returns the status of the inlet. `p` is a pointer to a `CellStatus` structure. The status parameter `cstat_t` is extended with the values `Extended` and `Retracted`.

`ACC_INQUIRY`

An inquiry command is sent to the jukebox and the inquiry data are returned. `p` is a pointer to a `inquiry` structure:

```
struct inquiry {
 unsigned char  i_type;       /* Peripheral device type */
 unsigned char  i_r1[3];      /* Removable media bit and others */
 unsigned char  i_xlen;       /* Extension length */
 unsigned char  i_r2[3];      /* Reserved */
 unsigned char  i_vid[8];     /* Vendor ID */
 unsigned char  i_prodid[16]; /* Product ID */
 unsigned char  i_frev[4];    /* Firmware revision level */
};
```

`ACC_MOVE`

A medium is moved from one element to another. `p` is a pointer to a `MovDsc` structure:

```
struct MovDsc {
```

```
 adr_t  trs;      /* Should have a valid or "Any" holder */
 adr_t  src,dst;
 side_t side;
};
```

The medium is moved from `src` to `dst` using the transport element `trs`. The medium is rotated according to `side`.

`ACC_MOVE_INLET`

The inlet is extended or retracted. `p` is a pointer to a `MovInlet` structure.

```
struct MovInlet {
      adr_t  trs;      /* Transport element address CHM */
      adr_t  adr;      /* source-adr=destination-adr=EEP */
      int    eepos;    /* extend/retract inlet */
      }
```

Values for the parameter `eepos` are:

```
#define SCMOVE_EXTENDEEP   0x40  /* extend the entry/exit port */
#define SCMOVE_RETRACTEEP  0x80  /* retract the entry/exit port */
```

`ACC_PARK`

The first transport element is moved to the specified storage element. `p` is an `unsigned char` and specifies the relative address of the storage element.

`ACC_PHOME`
`ACC_PINLET`

The first transport element is moved to the first or last storage element respectively.

`ACC_RESET`

ASCSI rezero command is sent to the device.

`ACC_SENDCMD`

This is a transparent SCSI interface; it should be used by Siemens software only. `p` is a pointer to a `sendcmd_t` structure:

```
typedef enum { S_IN=2, S_OUT=4, S_INOUT=6, S_NO=1 } direc_t;
typedef struct {
    unsigned char  s_cmd[12]; /* SCSI command descriptor block */
    unsigned char *s_buf;     /* Data buffer address */
    int            s_len;     /* Length of data buffer */
    direc_t        s_direc;   /* Direction of data transfer */
    int            s_status;  /* 4 byte status returned from driver */
    int            s_spare;   /* unused */
} sendcmd_t;
```

If illegal parameters are detected by the driver, `EINVAL` is returned. If the SCSI command has been executed successfully, `ioctl` returns 0 and errno is not set. Note that `s_status` may not be 0 in this case. If the command has not been executed successfully, `ioctl` returns –1 and errno is set to an appropriate value (see the section on **DIAGNOSTICS**).

The 32-bit status code `s_status` is defined as follows:

KKKKKKKK AAAAAAAA SSSSSSSS CCCCCCCC

K SCSI "sense key"

A SCSI "additional sense code"

S SCSI status returned to host adaptor (see `<sys/ios_cmd.h>`)

C Host adaptor error code (see `<ios_conf.h>`)

`ACC_SENSE`

Provides information on the last error that occurred. `p` is a pointer to a `dkerror` structure (see `sys/dkio.h`):

```
struct  dkerror {
      int     dke_error;     /* Error code */
      time_t  dke_time;      /* Error time */
      int     dke_resid;     /* Not implemented */
      int     dke_eccdata;   /* Not implemented */
      int     dke_errstat;   /* Error status */
      u_char  dke_evalid;    /* Valid Bit (information byte valid) */
      u_char  dke_embytes;   /* Number of valid more bytes */
      u_char  dke_emstart;   /* Start addr of more bytes */
      u_char  dke_spare;
      int     dke_errblk;    /* Extended Sense information bytes */
      u_char  dke_emore[16]; /* Extended Sense more bytes */
      int     dke_int[10];
};
```

Information on an error can only be queried once. If no errors have occurred since the last time `ACC_SENSE` was called, all elements of the `dkerror` structure are set to 0.

`ACC_TYPE`
: This returns the vendor ID from the inquiry string of the jukebox. `p` is a char pointer, which points to a string of at least 9 bytes in length.

DIAGNOSTICS

The following error codes are generated by the device driver `ioctl` interface:

`EBADF`
: The specified source of a move or exchange command was empty (corresponds to `ERR_SRC` in `acc/space.c`).

`ENOSPC`
: The specified destination of a move or exchange command was occupied (corresponds to `ERR_DST` in `acc/space.c`).

`ENOMEM`
: Timeout on import/export operation (corresponds to `ERR_INLET` in `acc/space.c`).

`EXDEV`
: The jukebox door has been opened; the status of the elements may have changed (corresponds to `ERR_DOOR` in `acc/space.c`).

`EINVAL`
: The `ioctl` code is invalid or not supported for the specified jukebox.

`ENXIO`
: Input parameters are out of range (illegal element address, illegal rotating code).

`EIO` An error has been reported by the jukebox; see `sjuke(7)` for information about hardware error reporting.

NOTES

Not all features are supported by all types of jukebox devices; see Hardware documentation for details.

This interface only supports jukeboxes with the SCSI device type "Medium Changer Device". Individual jukeboxes are operated using other interfaces [e.g. DAT Autoloader using the SCSI tape interface, see `stape(7)`].

FILES

`/dev/ios0/rsjuke`*`ppu`*

SEE ALSO

`devname(7)`, `sjuke(7)`, `stape(7)`, `autoconf(8)`, `jbsetup(8)`.

System Administrator's Reference Manual

adp(7)
RM400 only

NAME

`adp` - interface to the Adaptec ADP174x SCSI host adapter

SYNOPSIS

`/dev/ios0/scon` *xx*

DESCRIPTION

The EISA/IOS subsystem of the RM400 consists of up to four Adaptec ADP174x SCSI host adapters (`ios0/scon`*xx*) and SCSI device files [refer to `sdisk`(7) and `stape`(7)]. It is connected to the system via the EISA bus.

`ios0/scon`*xx* provides access to the Adaptec ADP174x SCSI host adapter through the device files in the `/dev/ios0` directory. The *xx* in the name of the SCSI host adapter stands for the logical number of the host adapter with an offset of 4. This is always specified as two decimal digits in the range 04 to 07.

MAJOR-MINOR DEVICE NUMBER

The (32-bit) major-minor device number is defined as follows:

`000`*MMMMM MMMMMM*`00 00000000 000000`*yy*

M The major device number.

y The logical number of the host adapter on the EISA bus.

IOS DESCRIPTION

Up to 4 SCSI host adapters of the Adaptec AHA174x family can be installed on RM400 systems. A mixture of both host adapters with a single-ended interface (ADP1740A) and host adapters with a differential interface (ADP1744) can be used here. Each Adaptec ADP174x SCSI host adapter supports a SCSI bus with up to seven SCSI devices.

IOS CONFIGURATION

The Adaptec ADP174x SCSI host adapters are identified and initialized when the system is loaded.

The SCSI-ID of the SCSI host adapter is fixed to 7.

However, it is important that no device is connected to the SCSI bus with the same SCSI-ID as the bus.

A unique logical device number is assigned to the individual host adapters when they are first recognized, which enables additional host adapters to be added without the device numbers of the devices already connected to the existing host adapters being changed.

In general, the device file for the SCSI host adapter is not used. The character special device is provided instead as a debug entry point to allow manipulation of the host adapter.

DIAGNOSTICS

Messages are output in the following format for the Adaptec ADP174x SCSI host adapter:

```
ios0/scon04: next_lpb != get_lpb
```

where:

`ios0/scon04:`
The Adaptec ADP174x SCSI host adapter where the error occurred. In this case, the error occurred on the host adapter at the EISA bus with logical number 0.

FILES

`/dev/ios`*b*`/scon`*xx*

SEE ALSO

devname(7), sdisk(7), stape(7).

System Administrator's Reference Manual

aio(7)

NAME

aio - primitive asynchronous I/O facility

SYNOPSIS

```
#include <sys/file.h>
#include <sys/ioctl.h>
#include <sys/async.h>
```

DESCRIPTION

NOTE:

The primitive asynchronous I/O facility is only available if the package "pkg.specaio" has been installed.

The aio facility provides processes with a primitive way of performing asynchronous read operations from, and write operations to a particular raw disk device. A process can start an I/O operation and instead of blocking until the I/O completes, the process can continue running. At some point in time, it is expected that the process will request the state of all I/O requests previously sent.

The process uses the same devices that it would normally use for raw disk I/O, however it must use the ioctl(2) mechanism to actually send the I/O request and to receive the state of previously sent requests.

It is also possible to have asynchronous notification of completed aio requests (which eliminates the need to continuously poll for completed requests, but generates a notification for each request). If this is enabled when a request is issued, then when that request finishes, a SIGEMT signal will be sent to the process.

The 4 ioctl commands that apply to aio are as follows:

DKIOCAIOVERS
: Return the version of the aio interface.

DKIOCMLOCK
: Lock a portion of memory for I/O transfer to/from disk.

DKIOCASTRT
: Start (send) an asynchronous I/O request.

DKIOCASTAT
: Return the state of previous requests.

The fcntl flag that apply to aio is:

FRAIOSIG
: Cause asynchronous notification of completed requests.

A very general algorithm describing the proper use of asynchronous I/O is given below:

1) The process can determine whether the aio interface is supported for a particular file descriptor using the DKIOCAIOVERS ioctl command.
2) The process allocates buffer space for the transfer of data to and from the disk.
3) It can optionally set the FRAIOSIG flag for asynchronous notification.
4) It can optionally lock the I\O buffer via the DKIOCMLOCK ioctl command.
5) The process now issues (several) I/O requests using the ioctl command DKIOCASTRT and then retrieves the state of those requests using the DKIOCASTAT ioctl command.

The DKIOCAIOVERS ioctl command should be used to determine whether aio is supported on the file descriptor and whether the kernel aio interface matches the interface used in the user program.

A process may decide to lock an area of memory which the device drivers will access to transfer the data for each request. This is done to keep those pages from being paged out while I/O is being performed. Memory is locked via the DKIOCMLOCK ioctl command:

```
ioctl(fd, DKIOCMLOCK, &memlock);
struct asyncmlock memlock;
```

The `struct asyncmlock` specifies the start (virtual) address of the memory to lock and the number of bytes to lock. Once locked, the locked portion of memory cannot be unlocked or changed. Only one request to `DKIOCMLOCK` is permitted during a process's lifetime.

If the memory is not locked using `DKIOCMLOCK`, then the kernel will handle the locking and unlocking of the memory specified in each request.

The `DKIOCMLOCK` command is included for downwards compatibility only. Memory should be locked and unlocked using the `MC_LOCK` and `MC_UNLOCK memcntl` commands.

The `FRAIOSIG` flag can be set if asynchronous notification of completed requests is desired.

To issue an I/O request, a process should use the `DKIOCASTRT ioctl` command:

```
ioctl(fd, DKIOCASTRT, &reqbuf);
struct areqbuf reqbuf;
```

The `struct areqbuf` specifies the command to be performed, the locations on disk and in core to transfer to/from, and the size of the transfer. The command may be either `AU_READ`, `AU_WRITE`, or `AU_ORDWRITE`. The difference between `AU_WRITE` and `AU_ORDWRITE` is that the latter, called an *ordered write*, causes all subsequent writes to physically occur only after the ordered write request has finished. (Normally, when a write request is issued, it is optimally sorted into the disk drive sort queue; e.g. the order of physical writes is independent of the order sent.) The size of the transfer *must* be an integral number of disk sectors and must be small enough to be handled in a single I/O request to the device.

The address of the user memory buffer has the same alignment requirements as normal reads and writes to the raw device. For most devices, word alignment is sufficient.

The size of a disk sector and the maximum permissible request size are contained in `struct dktype`. The `dktype` structure can be obtained from the driver using the `DKIOCGETTYPE ioctl`.

```
#include <sys/dkio.h>
ioctl(fd, DKIOCGETTYPE, &dkt);
struct dktype dkt;
```

If the `AU_ORDWRITE` command is used, it is up to the process to order the requests for optimum disk operation. *Note*: Incorrect use of ordered writes can severely reduce disk performance.

If no internal `aio` buffers are available, then the call will fail with the error `EWOULDBLOCK`.

To retrieve the state of completed I/O requests (for up to the number of requests specified by `MAXSTATUS`), a process uses the `DKIOCASTAT ioctl` command:

```
ioctl(fd, DKIOCASTAT, &iostat);
struct asyncstatus iostat;
```

The `struct asyncstatus` will be filled out by the `aio` driver and consists of a count of the number of requests for which status is being returned (if no requests have finished, then the count will be zero), and a data structure (`struct aiostat`) for each of those completed requests. The `aiostat` data structure consists of the error the driver returned for that particular request, and three fields that can be used to identify the request. These are the size of the request sent, its in-core memory address (as taken from the `struct areqbuf` argument to `DKIOCASTRT`), and a caller supplied value, `ioref`. The value of `ioref` will always be identical to the value of `au_ref` in the original request. The `au_ref|ioref` value may be any value containable in a character pointer.

Note:

> Please note the internal restriction on the maximum possible value that can be returned by parameter `iobsize` within structure `aiostat`.

The datatype of `iobsize` restricts the maximum number of bytes to transfer in an asynchronous I/O request.

The following example definitions (from `<sys/async.h>`) describe the `ioctl` commands and related data structures necessary for performing `aio`.

```
/* AIO version
 *  Return the version number of the current interface
```

```
 */
#define AIOVERSION      2
#define DKIOCAIOVERS    _IOR('S', 4, long)
/* Starting I/O (DKIOCASTRT)
 *
 *      'S' and '1' define a unique command number. Using
 *      _IOW specifies that the parameters are for input only.
 *
 *      The parameter passed back to SVR4 is a struct areqbuf.
 */
typedef struct  areqbuf {
 long    au_cmd;         /* AU_READ, etc */
 long    au_daddr;       /* destination on disk */
 char    *au_maddr;      /* (virtual) memory address */
 long    au_size;        /* bytes to transfer */
 char    *au_ref;        /* caller specified reference */
} AREQBUF;
/* command bits
 * Note:  AU_ORDWRITE is treated as AU_WRITE in the kernel
 *        They exist as separate flags for user convenience only.
 */
#define AU_READ         01      /* read request */
#define AU_WRITE        02      /* unordered write request */
#define AU_ORDWRITE     04      /* ordered write request */
#define AU_CMDMASK      07      /* mask of command bits */
/* The disk block to transfer to/from is in the lower 3 bytes
 * of au_daddr, the upper bytes are ignored by SVR4.
 */
#define ADBLK(A)        ((A) & 0xffffff)
#define DKIOCASTRT      _IOW('S', 1, AREQBUF)
/* Getting I/O Completion Status (DKIOCSTAT)
 *
 *      'S' and '2' define a unique command number.  Using
 *      _IOR specifies that the parameters are for output only.
 *
 *      The parameter passed back from SVR4 is a struct asyncstatus.
 */
typedef struct aiostat
{
 short   iostatus;      /* I/O completion status for request */
 short   iobsize;       /* verification (from AREQBUF dbsize) */
 char    *iomaddr;      /* verification (from AREQBUF dbmaddr)*/
 char    *ioref;        /* return caller specified reference */
} IOSTAT;
#define MAXSTATUS       15
typedef struct asyncstatus
{
 long    acount;             /* # of requests being returned */
 IOSTAT  astatus[MAXSTATUS]; /* completion status per request */
} ASYNCSTATUS;
#define DKIOCASTAT      _IOR('S', 2, ASYNCSTATUS)
/* Locking Memory for I/O (DKIOCMLOCK)
 *
 *      'S' and '3' define a unique command number. Using
 *      _IOW specifies that the parameters are for input only.
 *
 *      The parameters are passed to SVR4 as a struct asyncmlock.
 */
typedef struct asyncmlock
```

```
{
 char    *avaddr;        /* starting virtual address */
 unsigned asize;         /* size of area to be locked */
} ASYNCMLOCK;
#define DKIOCMLOCK      _IOW('S', 3, ASYNCMLOCK)
/* This is a new I/O error code - it really belongs in errno.h.
 * However, since it is AIO specific, it is here.
 */
#define EIORESID        500     /* block not (fully) transferred */
#ifdef  _KERNEL
#ifdef  i386
struct  aiomap
{
 caddr_t oldaddr; /* actual user address where 1I/O started */
 caddr_t newaddr; /* kernel address I/O job mapped to */
 int npgs;        /* number of pages of I/O job */
 u_int   phys1;   /* physical byte address of I/O job in page1 */
 u_int   phys2;   /* physical byte address of I/O job in page2 */
                      /* phys2 is always physically page aligned */
};
#endif
#endif
#endif /* _SYS_ASYNC_H */
```

DIAGNOSTICS

All errors except one (`EIORESID\fR) are the same as returned via the normal I/O system calls, such as read`, `write`, `ioctl`, etc. See the relevant system call (2) manual pages for information on these errors. Since `EIORESID` is specific to `aio`, it is defined in the system header file `<sys/async.h>` (rather than in `<errno.h>`) and is described below:

`EIORESID`

The driver accepted and queued the request; however, there was a subsequent error which caused the I/O to fail before the request finished. Some or none of the actual I/O took place.

NOTES

Using ioctls for I/O is not fun!

Currently, there is no portable way to determine the size of a disk sector on the underlying hardware.

FILES

`/dev/ios?/rsdisk*`

`/dev/vd/rvdisk*`

SEE ALSO

`fcntl(2)`, `ioctl(2)`, `sdisk(7)`, `vdisk(7)`.

System Administrator's Reference Manual

aio_ISTD(7)

NAME

aio_ISTD - Industry Standard (ISTD) asynchronous I/O interface

SYNOPSIS

```
#include <sys/async.h>
#include <sys/kaioapi_pyr.h>
```

DESCRIPTION

In addition to the existing asynchronous I/O interface [SPECAIO, DC/OSx, see aio(7)], the operating system supports a new set of asynchronous I/O interface functions. It is called the Industry Standard asynchronous I/O emulation facility (ISTD-AIO), loosely based on a subset of USL 4.2MP/P1003.4a.

The following library functions are part of this ISTD-AIO interface.

aioinit()
: Initialization function.

aiostat()
: Returns information indicating whether or not the ISTD-AIO interface is supported by the current kernel version.

aioread()
: Asynchronous read I/O request.

aiowrite()
: Asynchronous write I/O request.

aiopoll()
: Poll for outstanding asynchronous I/O requests.

For more details, see the manual pages for these functions. All functions are part of the libspecaio.a library. They use the existing SPECAIO interface [see aio(7)] with some extensions and a new special poll system call. The restrictions of the SPECAIO interface are also valid for ISTD-AIO.

Before ISTD-AIO I/O requests can be submitted, the aioinit() function has to be called once.

To make sure that the ISTD-AIO interface is supported by the operation system, the application has to call the aiostat() function.

A process can start an I/O operation [aioread() or aiowrite()] and instead of blocking until the I/O completes, the process can continue running. At some point in time, it is expected that the process will request the state of all I/O requests previously sent. This has to be done with the aiopoll() function.

The I/O requests and responses are defined in the ISTD-AIO specific aiocb structures.

The ISTD-AIO aiocb structures must be allocated by the application to manage its own AIO requests.

The aiocb is passed to the system with the aioread() or aiowrite() function.

After the aiocb control block is passed to the system, it belongs to the system. The address space of this aiocb control block must not be modified or released by the application until the aiocb is returned by the aiopoll() function.

The aiocb is structured as follows:

```
typedef volatile struct aiocb aiocb_t;
struct aiocb {
int      aio_fildes;     /* file descriptor */
void     *aio_buf;       /* buffer addr*/
size_t   aio_nbytes;     /* size of request */
ulong_t  aio_pad1[1];
off_t    aio_offset;     /* file offset */
ulong_t  aio_pad2[1];
```

```
off_t    aio_flag;       /* flag bits */
void     (*aio_notify)(aiocb_t *); /* notification func */
void     *aio_key;       /* retrieval key */
int      aio_errno;      /* status of IO */
aiocb_t  *aio_next;
ulong    aio_pad3[4];
      };
```

`aio_fildes`: The process uses the same devices that it would normally use for raw disk I/O.

`aio_buf` contains the source or target buffer address in the address space of the application.

`aio_nbytes` is the number of bytes to read or write. It has the same meaning as described in `aio(7)` for the `au_size` member of struct `areqbuf`.

`aio_offset` must be the block offset (unit is bytes) relative to the beginning of the file.

`aio_flag` must be zero (that means `AIO_NOHOLD`). `AIO_SEQ` is not supported.

`aio_notify` must be zero. Asynchronous notification is not supported.

`aio_key`: This is a user-specified key, which can be used to retrieve and sort the completion status of previously issued requests. This key is equivalent to the `au_ref` field in the `areqbuf` struct [see `aio(7)`].

`aio_errno`: This field reflects the status of the actual I/O request. It is set to the value `EINPROGRESS`, when the request is started and has not yet finished.

`aio_errno` is set to zero when the request has finished without error, otherwise it contains the error return-code. When the request has finished without error, then in addition to the `aio_errno` member, the `aio_nbytes` member is set to the number of bytes actually written or read. The application must also test `aio_nbytes` to ensure that all data has been read or written.

`aio_next` must be set to zero when submitting a new asynchronous I/O request.

When calling `aiopoll()`, this function returns the address of the first of a set of `aiocb` structures of completed I/O requests, chained together in the `aio_next` field. Zero indicates the end of the chain.

The `aio_pad*` members are reserved for future extensions. They should be set to zero.

Finally, the status structure returned by the `aiostat()` request is given below:

```
typedef struct kaioapi_aiostat {
       int     st_flags;
} aiostat_t;
```

The file `kaioapi_pyr.h` contains the `ISTD-AIO` specific data definitions.

Outstanding asynchronous I/O requests impact and are also impacted by other system calls in the following ways:

For `fork()`, no asynchronous I/O is inherited.

Before the `exec()` syscall can be performed, the system waits until all outstanding asynchronous I/O requests initiated by the caller are completed.

When closing a file descriptor, the system waits until all outstanding asynchronous I/O requests for this file descriptor are completed.

FILES

`/dev/ios?/rsdisk*`

`/dev/vd/rvdisk*`

SEE ALSO

`fcntl(2)`, `ioctl(2)`, `aioinit(3X)`, `aiopoll(3X)`, `aioread(3X)`, `aiostat(3X)`, `aiowrite(3X)`, `aio(7)`, `sdisk(7)`, `vdisk(7)`.

System Administrator's Reference Manual

aio_suspend_n(7)

NAME

aio_suspend_n - optimized aio_suspend function

SYNOPSIS

```
#include <sys/aio.h>
int aio_suspend_n(aiocb_t * list[],
  int nent,
  const struct timespec *timeout,
  int waitfor);
```

DESCRIPTION

The aio_suspend_n() function is an optimized version of the ABI AIO aio_suspend() library function. It is not part of the MIPS ABI interface definition, but is included as a Reliant UNIX specific enhancement in the libabi.so library.

Refer to the abiaio(7) manpage for a general description of the ABI AIO interface, the aiocb_t data structure and other structures relevant for ABI AIO.

The *list*, *nent* and *timeout* function parameters have the same meaning as for the aio_suspend() function [see abiaio(7)].

The *list* argument is an array made up of pointers to aiocb_t request structures. *nent* identifies the number of elements in the array. Each aiocb_t request structure indicated in this array must be defined in accordance with the descriptions of the aio_write(), aio_read() or lio_listio() functions [see abiaio(7)]. All pointers in the *list* array must point to valid AIO requests. NULL pointers are permitted but are ignored.

The following restriction must also be carefully observed: None of the AIO requests may use the callback mechanism.

The *timeout* parameter must be either NULL or point to a timespec structure that has a timeout time specified. If NULL is specified, then no timeout handling is performed (benefits throughput). The aio_suspend_n() function call returns to the calling process if either the timeout time has elapsed or the specified number of completed request messages are available.

The *waitfor* parameter specifies the number of AIO requests that should be ready before the function returns to the calling process. The value must be greater than 0 and less than or equal to *nent*.

aio_suspend_n() ends under any of the following conditions:

- An error situation is detected
- The completion messages for *waitfor* from the *list* array are available
- The timeout time has elapsed
- A signal has interrupted the function

RETURN VALUE

aio_suspend_n() returns a list of addresses of completed requests as well as the number of completed requests. The number is reported as a return value of the function. The aiocb_t addresses of the completed AIO requests are copied to the process address space using the *list* address.

If the function was interrupted by a signal or because the timeout time has elapsed, the actual number of completed requested at the time of the interrupt is returned. If there are no completed requests available, aio_suspend_n() returns –1 as a return value and errno is set to either EINTR or EAGAIN.

If an error is discovered, the return value is –1. The contents of the *list* array may have been changed in this case. errno contains additional information.

ERRORS

If `aio_suspend_n` returns –1, `errno` contains one of the following errors:

`EINVAL`
: The specified device is not open.

: The value of *list* is `NULL`.

: *nent* <= 0 or greater than the configured maximum possible number of AIO requests.

: The timeout time is specified incorrectly.

: One or more of the specified request addresses are invalid.

: Parameter *waitfor* is <= 0 or > *nent*.

`ENOMEM`
: No free memory available.

`EFAULT`
: Error copying data from or to the user address space.

`EAGAIN`
: The timeout time has elapsed and no completion message is available.

`EINTR`
: An interrupt occurred and no completion message is available.

SEE ALSO

`abiaio(7)`.

System Administrator's Reference Manual

bootvariable(7)
RM600 only

NAME

`bootvariable -` environment variables for boot2

DESCRIPTION

The system boot is specified in greater detail with the help of boot variables. The boot command is called as follows:

`boot` [`–f` *bootstring*] [*bootparameters*]

bootstring
defines the device from which Reliant UNIX is loaded.

bootparameters
can be specified if required and define values interpreted by the kernel during operation (e.g. `rootdev`, `swapdev`, and `bootflags`).

Saving boot variables

With operating system version V5.43 or later, boot variables are stored and evaluated independently of the boot mode. The boot mode is specified by the bootstring defined in CSI/HIOS-NVRAM.

Logical boot mode:

If the bootstring is entered in CSI/HIOS-NVRAM in logical format, booting takes place in logical mode. In order to be able to boot the root disk in logical mode, the parameters for this disk must have been entered in the root partition list in CSI/HIOS-NVRAM during installation. With logical booting, the boot variables are evaluated as follows.

1. Acceptance of values from the root partition list in CSI/HIOS-NVRAM (boot parameters `rootdev` and `swapdev`)
2. Acceptance of boot parameters defined in `/etc/default/boot`
   (all boot parameters except `rootdev` and `swapdev`)
3. Interactive entry of boot variables

Values at level 3 overload values at level 1 or 2.

Physical boot mode:

If the bootstring is entered as a physical bootstring [format `sd(x,x,x)`], the entire booting process takes place in physical boot mode (compatible with the old boot mode). The boot variables can then be accepted and/or entered in different ways:

1. Acceptance of values defined in CSI/HIOS-NVRAM (bootstring only)
2. Acceptance of values defined in `/etc/default/boot`
3. Interactive entry of values (in interactive boot mode only)

The following general rules apply:

Values at level *n*+1 overload values at level *n*.

Examples:

If the bootstring is defined in `/etc/default/boot` in automatic mode, the value defined in the NVRAM is overloaded.

In interactive boot mode, the values entered in the command line overload all other values of the boot variables.

Setting the bootstrings:

The value set for the bootstring can be queried in BDM using the `ra` command. The

`wn 0 bootstring`

command is used to define a new value.

Examples:

`wn 0 sd(0,0,0)`
sets the value for the physical bootstring to `sdisk(0,0,0)`

`wn 0 sdisk053s0`
sets the value for the logical bootstring to `sdisk053s0`

Definition of the bootstring

The bootstring defines the device from which Reliant UNIX is to be loaded.

A description of the structure of the RM600 bootstring follows. The boot string has one of the formats described below, depending on the boot mode:

Logical boot mode:

The bootstring notation corresponds to UNIX disk name notation [see `sdisk(7)`].

Example: `sdisk053s0/unix`

Physical boot mode:

The bootstring has the following format:

xx (*a*,*b*,*c*) *file*

This means:

xx Driver ID:

`sd` SCSI disk

`cd` CD-ROM

a Channel ID

b SCSI ID

c Partition or file number

file Name of the file to be loaded

Meaning of the values in brackets for selecting SCSI devices:

a = Channel ID

Bit 0 – 4
logical SCSI bus number

The values are defined as follows:

CSI-SCSI
Channel ID = 0.

DCS The channel IDs are assigned to the DCS-SCSI channels, in ascending order of the DCS slot IDs.

HIOS The channel IDs are assigned in accordance with the location of the SCSI channels in the system cabinet.

	Channel ID	SCSI backplane	SCSI bus
RM600-E20 + RM600-E60	0	1	0
	1	1	1
	2	1	2
RM600-E60	3	0	0
	4	0	1
	5	0	2

Bit 3 – 15
free

b = Selection of the SCSI device on the SCSI channel

Bit 0 – 2
LUN, Logical Unit Number (currently = `0`)

Bit 3 – 6
SCSI Target ID

Bit 7 – 15
free

c = Selection of the partition number or file number for tapes

In the case of BOOT2 for SCSI tapes and video tapes, this parameter determines the number of the file on the tape. 0 is the number of the first file.

The value to be defined is assigned to the boot variable `bootfile` in `/etc/default/boot`.

Example: `BOOTFILE="sd(0,40,0)unix"`

Definition of boot variables in /etc/default/boot

The file in the root partition named `/etc/default/boot` saves boot variables evaluated using BOOT2. Bootstrings and boot parameters can be defined using `/etc/default/boot`. `/etc/default/boot` is an ASCII file and can be edited under Reliant UNIX using any editor.

Defining the boot parameters

Up to 8 boot parameters can be defined in `/etc/default/boot`, which are then interpreted by Reliant UNIX at startup. The boot parameters (*x* = 0 – 7) are assigned to the boot variables `BOOTP`*x*.

Examples:

```
BOOTP2="bootflags=0x40"
```

The root disk is booted as a mirrored root disk.

```
BOOTP3="initarg=-s"
```

Reliant UNIX boot up to single user mode is executed.

Interactive boot variable entry

If the `boot` command is called without boot variables, the current boot variables are used.

If boot variables are specified in the `boot` command line, these boot variables overload the current boot variables.

Examples:

```
boot –f /unix.org
```

BOOT2 loads `unix.org` from the active root disk and passes on the boot parameters defined in `/etc/default/boot`.

```
boot initarg=-s
```

BOOT2 loads Reliant UNIX from the root disk defined as active in the root partition list and boots it in singleuser mode.

SEE ALSO

`boot2`(8), `bootflags`(8).

System Administrator's Reference Manual

ccfg(7)
RM600 only

NAME

`ccfg -` controller configuration

DESCRIPTION

The `/dev/ccfg` file is a character-oriented pseudo device used to address the controller configuration driver `ccfg`. `ccfg` assigns logical numbers to the RM600 controllers. If there are several multibus processors on a multibus, `ccfg` distributes the load of the I/O jobs on the multibus processors.

IOCTL calls

The `ioctl` calls are only released for internal use and are therefore not described here.

FILES

`/dev/ccfg`

SEE ALSO

`controller(4)`, `ldcontroller(8)`, `mkcontroller(8)`, `modcontroller(8)`, `showconf(8)`.

System Administrator's Reference Manual

cit(7)
RM600 only

NAME

`cit` - CSI-TTY interface

DESCRIPTION

`cit` is a device file that supports the connection to the CSI controller over the 6 serial V.24 interfaces.

`cit` is a STREAMS-based driver, and its major number is 128. Acceptable minor numbers are in the range 0 to 5 for interfaces with no modem functionality, and 256 to 261 for those with modem functionality. Interface signals S1, S2, M1, and M5 are evaluated.

For a description of the system call interface of the `cit` device file, please see `termio(7)` and `termios(3C)` .

The V.24 interfaces should be used as follows:

Console
: `/dev/console` has minor number 1. The 1st port (9-pin) from the top on the CSI is addressed using this node.

Modem port
: `/dev/modem` has minor number 256. The 2nd port (25-pin) from the top on the CSI is addressed using this node.

UPS `/dev/term04` has minor number 4. The port to the left of the chassis (9-pin) is addressed using this node.

Radio clock
: `/dev/term05` has minor number 5. The port to the right of the chassis (9-pin) is addressed using this node.

Free `/dev/term02` has minor number 2. `/dev/term03` has minor number 3. The 3rd and 4th ports from the top (9-pin) on the CSI are addressed using these nodes.

FILES

```
/dev/console
/dev/ttyc1
/dev/term01
/dev/term02
/dev/term03
/dev/term04
```

SEE ALSO

`termios(3C)` , `termio(7)` .

System Administrator's Reference Manual

clone(7)

NAME

`clone` - open any major/minor device pair on a STREAMS driver

DESCRIPTION

`clone` is a STREAMS software driver that finds and opens an unused major/minor device on another STREAMS driver. The major device number passed to `clone` during open corresponds to the `clone` driver and the minor device number corresponds to the target driver. Each open results in a separate stream to a previously unused major/minor device.

The `clone` driver consists solely of an open function. This open function performs all of the necessary work so that subsequent system calls [including `close(2)`] require no further involvement of `clone`.

`clone` will generate an `ENXIO` error, without opening the device, if the major/minor device number provided does not correspond to a valid major/minor device, or if the driver indicated is not a STREAMS driver.

NOTES

Multiple opens of the same major/minor device cannot be done through the `clone` interface. Executing `stat(2)` on the file system node for a cloned device yields a different result from executing `fstat(2)` using a file descriptor obtained from opening the node.

SEE ALSO

`log(7)` .

Programmer's Guide: STREAMS

System Administrator's Reference Manual

connld(7)

NAME

connld - line discipline for unique stream connections

DESCRIPTION

connld is a STREAMS-based module that provides unique connections between server and client processes. It can only be pushed [see streamio(7)] onto one end of a STREAMS-based pipe that may subsequently be attached to a name in the file system name space. After the pipe end is attached, a new pipe is created internally when an originating process attempts to open(2) or creat(2) the file system name. A file descriptor for one end of the new pipe is packaged into a message identical to that for the ioctl I_SENDFD [see streamio(7)] and is transmitted along the stream to the server process on the other end. The originating process is blocked until the server responds.

The server responds to the I_SENDFD request by accepting the file descriptor through the I_RECVFD ioctl message. When this happens, the file descriptor associated with the other end of the new pipe is transmitted to the originating process as the file descriptor returned from open(2) or creat(2) .

If the server does not respond to the I_SENDFD request, the stream that the connld module is pushed on becomes uni-directional because the server will not be able to retrieve any data off the stream until the I_RECVFD request is issued. If the server process exits before issuing the I_RECVFD request, the open(2) or the creat(2) system calls will fail and return –1 to the originating process.

When the connld module is pushed onto a pipe, messages going back and forth through the pipe are ignored by connld.

On success, an open of connld returns 0. On failure, errno is set to the following values:

EINVAL
: A stream onto which connld is being pushed is not a pipe or the pipe does not have a write queue pointer pointing to a stream head read queue.

EINVAL
: The other end of the pipe onto which connld is being pushed is linked under a multiplexer.

EPIPE
: connld is being pushed onto a pipe end whose other end is no longer there.

ENOMEM
: An internal pipe could not be created.

ENXIO
: An M_HANGUP message is at the stream head of the pipe onto which connld is being pushed.

EAGAIN
: Internal data structures could not be allocated.

ENFILE
: A file table entry could not be allocated.

SEE ALSO

streamio(7) .

Programmer's Guide: STREAMS

System Administrator's Reference Manual

console(7)

NAME

`console` - STREAMS-based console interface

DESCRIPTION

The `/dev/console` file is the system console and refers to an asynchronous serial data line originating from the system board.

The file `dev/contty` refers to a second asynchronous serial data line originating from the system board.

Both `/dev/console` and `/dev/contty` access the STREAMS-based console driver, which when used in conjunction with the STREAMS line discipline module `ldterm`, supports the `termio(7)` and `termios(3C)` processing.

FILES

`/dev/console`
`/dev/contty`

SEE ALSO

`crash(1M)`, `termios(3C)`, `ldterm(7)`, `termio(7)`.
Programmer's Guide: STREAMS

System Administrator's Reference Manual

cpt(7)
RM400 only

NAME

`cpt, bcpt -` Centronics interface control

DESCRIPTION

The `/dev/cpt` file is a special file that controls the Centronics interface for printers.

The `/dev/bcpt` file is a special file that controls the Centronics interface for printers in both directions (sending and receiving data). This interface conforms to IEEE 1284 and supports Compatibility Mode, Nibble Mode and Extended Capabilities Port (ECP) Mode. The mode is chosen automatically during negotiation phase, which is initiated after the `open()` system call to `/dev/bcpt`.

Only one `open()` is permitted at a time for both devices. An `open()` system call to `/dev/cpt` or `/dev/bcpt` while one of these devices is already open will return with error `EBUSY`.

If the printer device is powered down while transmitting data via the Centronics interface, the `cpt` driver sends a stream message `M_HANGUP` to the streams head.

IOCTL

Only one `ioctl()` is supported.

`TCSBRK`
: The argument is an int value. Wait for the output to drain. If the argument is 0, then send a break (zero valued bits for 0.25 seconds).

NOTES

The device waits for the output to drain before it closes. If any error occurs, device will close after some time (`cpt` closes on paper empty!).

FILES

`/dev/cpt`
`/dev/bcpt`

SEE ALSO

`termio(7)`.

System Administrator's Reference Manual

csimaint(7)
RM600 only

NAME

`csimaint -` access to the CSI maintenance function

SYNOPSIS

```
#include <sys/io/csi.h>
#include <sys/io/csimnt.h>
int ioctl(int fildes, int request, maint_ioctl_t *arg);
```

DESCRIPTION

The `/dev/csimaint` file enables you to access the maintenance function CSI (Central Services & Interfaces). This maintenance function can be used to query and change settings in the NVRAM of the CSI. You can also issue commands to the connected EIPs (Environment Interface Processor).

The commands available are described below in more detail.

The `ioctl` call expects the following parameters:

`fildes`
: File descriptor returned when the device node was opened.

`request`
: Device-dependent function code to which the value `MAINTIOCTL` must be assigned.

`arg` Pointer to a device-dependent parameter block with the following structure:

```
typdef struct maint_ioctl
{
 struct maint_cmd   *mi_cmd;
 void               *mi_dat_out;
 uint_t              mi_len_out;
 void               *mi_dat_in;
 uint_t              mi_len_in;
 int                 mi_herr;
 struct maint_stat  *mi_stat;
 int                 mi_merr;
} maint_ioctl_t;
```

The meaning of the individual fields is as follows:

`mi_cmd`
: Pointer to a completely full command block.

`mi_dat_out`
: Pointer to a buffer for transferring DMA data to the CSI.

`mi_len_out`
: Length of the output data. If no data is transferred, the value zero must be assigned.

`mi_dat_in`
: Pointer to a buffer to which data is to be transferred by the CSI.

`mi_len_in`
: Length of the input data. If no data is supplied, the value zero must be transferred.

`mi_herr`
: Error code as supplied by HL-MUCI (High Level Multibus II Communication Interface). A value of zero means that the command was sent successfully. A value other than zero implies an error occurred when sending the command. The contents of the status block are then undefined.

`mi_stat`
: Pointer to a status block as supplied by the CSI when it is acknowledged. Without an acknowledgement request, this parameter is not evaluated.

`mi_merr`
: Error code as supplied by MUCI (Multibus II Communication Interface). A value of zero means that the command was sent successfully. A value other than zero means that an error has occurred. The contents of the status block are then undefined.

The individual functions are now described in detail. The structures used `maint_cmd` and `maint_stat` are defined in the `csi.h` file. When describing the functions, only the fields whose values are evaluated are named.

Callback numbers

Telephone numbers and passwords for the automatic teleservice callback can be stored in the NVRAM of the CSI.

The command structure must be initialized as follows for this purpose:

```
mc_dev_no   = 0x00;
mc_flags    = CF_ACK|CF_DAT;
mc_cmd_code = M_CALLBACK_NUMBERS;
mc_modifier = 0x00;
```

The number of telephone entries can be specified via the substructure `s` of the union `u` in the command structure. The number (max. 18) is assigned to the `mc_length` field.

Telephone numbers and passwords must be stored in the `mi_dat_out` string as follows:

Byte00 – Byte09	Password for number 1
Byte10 – Byte29	Relevant telephone number
Byte30 – Byte39	Password for number 2
Byte40 – Byte59	Relevant telephone number

.

.

.

The password and the telephone numbers are ASCII characters and end with `0x00` if the password has less than 10 characters and the telephone number has less than 20 characters.

If the last valid character in the telephone number is "Z", this number is partially qualified. Before callback is possible, the caller is asked for the remaining characters.

Errors are reported via the `ms_error` field to the status structure `maint_stat`:

`ms_error & 0x02`	No data available
`ms_error & 0x04`	Too many entries

Status query

You can call up the following status information:

- General status information
- Power failure time
- RM600 configuration
- Console number

The command structure must be initialized as follows for this purpose:

```
mc_dev_no   = 0x00;
mc_flags    = CF_ACK;
mc_cmd_code = M_READ_STATUS;
```

Select the information you require using the values for the `mc_modifier` field (see above). The following values are possible:

`M_TASK_INFO`	General status information
`M_POWER_FAIL`	Power failure time
`M_B20_CONFIG`	RM600 configuration
`M_CONSOLE_NO`	Check console number

The required information is stored in the status structure. The following data is returned for each item of information stored:

`M_TASK_INFO`

Information can be accessed via the `rs00` structure of the `u` union in the status structure:

`ms_mem_size`	–	Size of the reserved maintenance memory in 4-Kbyte blocks.
`ms_cmd_q_len`	–	Length of the command list
`ms_ctrl_data[5]`	–	Offset with a "Solicited Message" `CSI_INIT`
`ms_slot_id`	–	Slot number of the port which receives the "solicited status messages"
`ms_port_id`	–	Number of the port which receives the "solicited status messages"

`M_POWER_FAIL`

The information can be accessed via the `rs07` structure in the `u` union of the status structure:

`ms_seconds \z \z|\z|\z|\z}\z|\z|\z|`

`ms_minutes`

`ms_hour` – Power_Fail time

`ms_day`

`ms_month`

`M_B20_CONFIG`

Information can be accessed via the `rs09` structure in the `u` union of the status structure:

`ms_board_count`	–	Total number of boards in system

`M_CONSOLE_NO`

Information can be accessed via the `rs0b` structure in the `u` union of the status structure:

`ms_console_no`	–	Device no. of the console (1–5)

Reading the boot parameters

The command structure `maint_cmd` for this function must be initialized as follows:

```
mc_dev_no   = 0x00;
mc_flags    = CF_ACK;
mc_cmd_code = M_READ_BOOT_PARAM;
mc_modifier = 0x00;   /* Parameters are read from the NVRAM */
```

```
mc_modifier = 0x01;   /* Parameters are read from the DRAM  */
```

The boot parameters are transferred to the `mi_dat_in` field of the parameter block (see above). They correspond to the structure described below:

```
struct boot_data {
    uint_t bd_boot_flags;
    char bd_boot_string[80];
    char bd_dump_string[80];
};
```

Writing the boot parameters

The command structure `maint_cmd` for this function must be initialized as follows:

```
mc_dev_no   = 0x00;
mc_flags    = CF_ACK|CF_DAT;
mc_cmd_code = M_WRITE_BOOT_PARAM;
mc_modifier = 0x00;  /* Parameters are written to the NVRAM  */
mc_modifier = 0x01;  /* Parameters are written to the DRAM   */
```

The boot parameters are expected in the `mi_dat_out` field of the parameter block. They correspond to the `boot_data` structure (see above).

EIP commands

The command structure `maint_cmd` for this function must be initialized as follows:

```
mc_flags    = CF_ACK;
mc_cmd_code = M_EIP;
```

The relevant EIP command is defined via the `mc_modifier` field of the `maint_cmd` structure (see above). The following functions can be executed using the specified commands:

- Define the EIP version
  `M_GET_EIP_VERS`
- Describe the LCD
  `M_WRITE_TO_LCD`
- Query the menu button
  `M_READ_PANEL_BUTTON`
- Change the color of the LED
  `M_SWITCH_LED`
- Request the status of a cabinet
  `M_GET_CABINET_STATUS`
- Determine the EIP test status
  `M_GET_EIP_STATUS`
- Determine the firmware release
  `M_GET_EIP_FW_REL`
- Request the BBU status
  `M_READ_BBU_STATUS`
- Request the USV status
  `M_READ_USV_STATUS`

The following applies for all commands:

The `ms_reserved` and `ms_error` fields of the status structure are used for error messages.

The following error codes are possible for `ms_reserved`:

`0x00` OK

`0x01` A field of the command is assigned an invalid value.

`0x02` The command requires "solicited data". No data was transferred.

0x03 The specified field mc_modifier is unknown.

0x04 The command is not valid for the current EIP version.

0x10 An internal firmware problem has arisen.

0xff The EIP command sent is not known.

The following error codes are possible for ms_error:

0x00 OK

0x01 An illegal command was sent to the EIP. This is due to an internal firmware problem or there is a breakdown in the V24 connection to the EIP.

0x02 The EIP could not send an answer within the timeout period. The EIP may not be functioning correctly.

- Define the EIP version

  The ms_port_stat field of the status structure contains the version:

  0x01 EIP-1

  0x02 EIP-2

- Describe the LCD

  This command can be used to write a piece of ASCII text to the LCD on the control panel. The LCD consists of two lines each with 16 characters.

  The length of the data to be displayed (<16) is assigned to the mc_modifier1 field whereas the line number (½) is assigned to the mc_modifier2 field. The column from which the text is to be output is coded in the following byte.

  The text is transferred to the mi_dat_out field in the parameter structure.

- Query menu button

  The menu button can have three different statuses:

  – Button was not pressed (0x00)

  – Button was pressed quickly (0x01)

  – Button was held down before being released (0x02)

  A timeout value is assigned to the mc_modifier field. The EIP waits the specified time if the button is pressed. The relevant status is returned in the ms_stat1 field of the status structure.

- Change the color of the LED

  This command is used to switch the color of the LED on the control panel between red and green.

  The color is determined via the mc_modifier1 field and is coded as follows:

  0x00 green

  0x01 – 0xff
  red

The following commands are issued specifically for a system cabinet. The number of the relevant cabinet must be assigned to the mc_modifier1 field. This field is a component of the eip structure in the u union of the maint_cmd command structure.

- Determine the cabinet status

  The status is coded as a bit vector in the ms_stat1 and ms_stat2 fields in the eip structure of the u union in the status structure. The following values are possible:

```
ms_stat1 & EIP_ACFAIL2               ACFAIL2 activems_stat1 & EIP_TEMP_FAIL
          TEMP_FAIL activems_stat1 & EIP_CABINET_FAN_FAIL      FAN_FAIL active
ms_stat1 & EIP_POWERSUPP_FAN_FAIL    FAN_FAIL in the power supplyms_stat1 & EI
P_ACFAIL1               ACFAIL1 activems_stat1 & EIP_POWER_ON              POW
ER_ON activems_stat1 & EIP_EIP_LOST              EIP_LOST activems_stat2 & EIP
_EIP_BAD               EIP2 with this no. not available
```

- Determine the EIP test status

  The test status is returned as a bit vector in the `ms_stat1` and `ms_stat2` fields in the `eip` structure of the `u` union in the status structure.

  The bits are assigned to different tests. A value of 0 means that the test ended correctly, whereas a value of 1 means that an error occurred. The following is a list of possible assignments:

  Bit 0: `E_ROM`, checksum error
  1: `E_RAM`, RAM test
  2: `E_NVR`, NVRAM test
  3: `E_PAN_CNTR`, access to control panel
  4: `E_PAN_DATA`, access to control panel
  5: reserved
  6: `E_CAN`, CAN controller test
  7: reserved
  8: `E_NOT`, testing the emergency power operation
  9: `E_FAN`, testing for fan malfunction
  10: reserved
  11: reserved
  12: reserved
  13: reserved
  14: `E_REL`, master and slave EIPs have different releases
  15: `E_NONE`, EIP2 with the specified cabinet number does not exist

- Determine the firmware release

  The version number is stored in the `ms_stat1` field in the `eip` structure of the `u` union in the status structure whereas the release number is stored in the `ms_stat2` release.

- Query the BBU status

  The type of BBU (Battery Backup Unit) used is coded in the `ms_port_stat` field in the status structure. The following assignment applies:

  `0x00` No BBU in the cabinet
  `0x01` BBU, type 3193.XX
  `0x02` External BBU, 5 KW
  `0x04` Power supply with integrated BBU, type 3204
  `0x08` – `0x80`
  Reserved for future BBU types

  In the case of BBU types 3193.XX and 3204, the status is returned in the `ms_stat1` field of the `eip` substructure in the status structure.

  The following bit assignments apply:

  Type 3193.XX:

  Bit 0: reserved, i.e. cannot be evaluated
  1: =0 battery charger is defective
  =1 battery charger OK

2: reserved
3: =0 DC cable not connected
=1 DC cable connected
4: reserved
5: reserved
6: =0 BBU operation (i.e. power failure)
=1 no BBU operation
7: reserved

Type 3204:

Bit 0: reserved, i.e. cannot be evaluated
1: =0 battery charger is defective
=1 battery charger OK
2: =0 battery empty
=1 battery OK
3: =0
4: reserved
5: reserved
6: reserved
7: =0 BBU operation (i.e. power failure)
=1 no BBU operation

For external BBU cabinets, the status is returned coded as follows (the following applies for bit = 0):

`ms_stat1`:

Bit 0: Network relay OFF (only test status)
1: Battery charger is defective
2: Battery depth discharging status reached
3: Max. battery charging status reached
4: Temperature error in the battery charger
5: Service switch ON
6: BBUs functional test
7: Breakdown in the line voltage at the battery charger

`ms_stat2`:

Bit 0: BBU 1 temperature error

1: BBU 2 temperature error
2: BBU 1 active
3: BBU 2 active
4: BBU 1 connected and backup OK
5: BBU 2 connected and backup OK
6: System backup for BBU 1 defective
7: System backup for BBU 2 defective

The `ms_stat3` and `ms_stat4` fields are the same as the `ms_stat2` field for the BBUs 3, 4, 5, and 6 in other cabinets.

- Query USV status

  This command can be used to determine the status of a connected UPS (uninterruptible power supply) of the type B40 or 9071.

  The status of the following signal lines is returned in the `ms_port_stat` byte.

  B40 type series:

  Bit 0: =0 USV active, power failure
  =1 USV not active
  1: =0 inverter off
  =1 inverter on
  2: reserved
  3: reserved
  4: =0 battery discharged
  =1 battery OK
  5: =0 inverter malfunction
  =1 no malfunction
  6: reserved
  7: reserved

  9071 type series:

  Bit 0: =0 USV not active
  =1 USV active, power failure
  1: =0 Bypass not active
  =1 Bypass active
  2: reserved
  3: reserved
  4: =0 battery OK
  =1 battery discharged
  5: =0 USV OK

=1 USV sum alarm

6: reserved

7: reserved

DIAGNOSTICS

If `ioctl` is executed successfully, the value zero is returned whereas if an error occurs, the value returned is –1. The relevant error code is contained in `errno`.

The following error codes are returned:

`EINVAL`
: An invalid function code is returned. □

`EFAULT`
: An error occurred when transferring parameters or data from the user address space to the kernel address space. □

`ENOMEM`
: There is insufficient space to buffer the output or input data. □

`EIO` HL-MUCI or MUCI returned an error. The output parameters `mi_herr` and `mi_merr` provide detailed information (see above).

FILES

`/dev/csimaint`

SEE ALSO

`eip2(7)`.

System Administrator's Reference Manual

devfs(7)

NAME

`devfs` - device file system

DESCRIPTION

The `/devfs` file system enables access to connected mass storage media on the basis of serial numbers or the `autoconf` name of the mass storage.

The files in `/devfs` are accessed for reading using standard system calls, for example `open(2)`, `mount(2)`, `read(2)` and others.

NOTES

The file system offers a file `info` under the mountpoint *mount_point* [cf. `mount(1M-devfs)`]. This file contains a list with an assignment of the names generated under the mountpoint and the name of the device in the `autoconf` tree.

For performance reasons, the file system uses an internal cache in the `autoconf` tree. This cache is halted by default for half a second. If the `autoconf` tree itself or its internal information is changed, these changes are only visible when the cache hold time has expired.

The cache hold time can be modified using the `devfscachevalid` variable in the `/etc/conf/pack.d/devfs/space.c` file. It may be useful to increase the hold time for systems that have a large number of SCSI devices. `timex ls -l` *mount_point* can be called to establish if it makes sense to increase the cache hold time for a particular system. The `devfscachevalid` variable can be modified during runtime using suitable programs [e.g. `crash(1M)`].

FILES

`/etc/conf/pack.d/devfs/space.c`
: contains the `devfscachevalid` configuration variable

SEE ALSO

`mount(1M)`, `mount(1M-devfs)`, `autoconf(8)`.

System Administrator's Reference Manual

devname(7)

NAME

devname - device naming conventions

DESCRIPTION

Specific conventions are used to assign device names. When a utility requires the *devname* to be used as a parameter (for example, dksetup), it requires the device name format described here. This *devname* format is not the same as the device node name or file system name for a device.

Mass Storage and I/O Name Features

The naming convention, described in the sections below, is applied to each current I/O mass storage subsystem. This convention has the benefit of simplicity of device-path names, consistency between the differing I/O subsystems and allows for future growth. Each name reveals the device's physical connection to the system as well as any specific attributes, such as the rewind device for tapes.

FEATURES

The Reliant UNIX naming convention has the following features:

Physical Association
The name describes the physical device to the user. It uniquely corresponds to the device, and does not change if I/O devices are later added to, or deleted from the system.

Large Configurations
The device name accommodates large system configurations with hundreds of devices.

Multiple I/O Buses
The device naming supports configurations with multiple I/O buses. It unambiguously includes the bus number. For RM600-xx computers, the native bus is the System XTEND Bus 0.

Multiple I/O Subsystems
The device naming scheme is consistent across multiple I/O subsystems.

Simplicity
The name used to describe the device should is as simple as possible, with as few physical device descriptors as necessary.

FILES

/dev/ios*b*/[r]stape*ppu*[c][h][n][o][v]
/dev/ios*b*/[r]sdisk*bcusp*
/dev/ios*b*/slp*xx*[r|c]

SEE ALSO

mtio(7), sdisk(7), sgen(7), sjuke(7), stape(7), autoconf(8).

System Administrator's Reference Manual

digi(7)

NAME

`digi -` serial interfaces offered by AccelePort *X*r 920 PCI boards

DESCRIPTION

AccelePort *X*r 920 PCI boards from Digi International Inc. offer a number of serial interfaces (V.24). These boards are supported by the `digi` driver from the `SIdigi` package.

Note: *X* is replaced in the actual model name by the number of available serial interfaces; thus, for example, the "AccelePort 2r 920-PCI" board has two serial interfaces.

Devices

The serial interfaces are accessed via devices, which are typically located under `/dev/term`. The `digi` driver has the major number 392, while the minor number is taken from the board number B (assigned by `ccfg`) and the port number P (beginning with 0) as follows:

Minor number = 128 * B + P (local device without modem control)

or

Minor number = 128 * B + P + 64 (device with modem control)

The command for enabling a device for port 0 on board 0 would then be as follows, for example:

```
mknod c /dev/term/digi0 392 0
```

There is also an administration device called `/dev/digi_adm` (minor number 12800) and a trace device called `/dev/digi_trace` (minor number 12801).

Administration

The boards (and the individual ports on them) can be administered via the `/dev/digi_adm` administration device. Commands are written in plain text to the device and the responses are read from there. The commands all have the form:

[*board_number*[**,***port_number*]]:*command*

without whitespace in each case and terminated by a newline character. If *board_number* and *port_number* are not specified, the command applies for all boards or all ports.

The following values are supported for *command*:

`report`
: Returns the status of the board. If *board_number* has been specified, a status line is output for each port. The port status line contains various variables with values, for example `port=0` for port 0. For an explanation, see below.

`load_bios`
: Loads the BIOS on the board (see below). The code is expected after the terminating newline character; if a *port_number* is specified, it is ignored.

`load_fepos`
: Loads the FEPOS firmware on the board (see below). The code is expected after the terminating newline character; if a *port_number* is specified, it is ignored.

`fast_baud`
: Enables alternative baud mapping (for high speeds, see below).

`normal_baud`
: Standard baud mapping (1:1) is set.

`use_dsr`
: The DSR signal is regarded as a carrier.

`use_cd`

The CD signal is regarded as a carrier.

`reset`
The board is reset and any *port_number* specified is ignored.

Once a command has been written to the administration device, the device should be read out. If successful, the device returns the required data or `OK`, or otherwise an error message.

Loading

The boards must first be loaded so that they can be operated, i.e. with a BIOS and the FEPOS firmware. Both are supplied by Digi International and can be located in the file system under `/opt/digi/load/xrbios.bin` or `/opt/digi/load/xrfep.bin`. The board works through the following states when the system is started:

RESETTING
Initiates a reset on the board (system startup or command).

RESET_FAILED
Board could not perform a reset.

RESET_OK
Board has performed a reset.

BIOS_LOADING
After RESET_OK: Board is loaded with BIOS.

BIOS_LOAD_READY
Loading of the BIOS is completed, BIOS initializes itself.

BIOS_FAILED
BIOS failed.

BIOS_OK
BIOS initialization completed successfully.

FEPOS_LOADING
After BIOS_OK: Board is loaded with FEPOS.

FEPOS_LOAD_READY
Loading of the FEPOS is completed, FEPOS initializes itself.

FEPOS_FAILED
FEPOS failed.

RUNNING
FEPOS initialization completed successfully. The board is operational.

A board can only be operated with the BIOS if it has the status RESET_OK; similarly it can only be loaded with FEPOS if it has the status BIOS_OK. The board is loaded by the administration device using appropriate commands (see above). Loading is only performed if the board has the correct status.

Example:

```
(echo :load_bios; cat /opt/digi/load/xrbios.bin) > /dev/digi_adm
```

Loads all boards with the BIOS if the status is RESET_OK.

The loading of the boards is handled at system startup by the `/etc/rc2.d/S10digi` rc script.

Modem control

The `digi` driver supports local devices (without modem control) as well as devices with modem control (for minor numbers see above in the section **Devices**).

A port can either be operated locally or with modem control at any one time. (Where an attempt is made to open both devices for a port at the same time, the second `open()` attempt produces an `EBUSY` error.)

Local devices
The CD, DSR and CTS input signals are ignored. An `open()` system call sets DTR and RTS to active. There is no hardware flow control.

In the case of a `close()` system call where `HUPCL` is set (in the termio structure), DTR and RTS are set to inactive.

Device with modem control

A blocking `open()` system call waits for the active CD input signal. The port works with hardware flow control via the RTS/CTS signals. If `CLOCAL` is not set in the termio structure, a hangup message is created if CD is inactive.

In the case of a `close()` system call where `HUPCL` is set (in the termio structure), DTR and RTS are set to inactive.

DSR uses the `use_dsr` command (see the section **Administration** above) to take over the CD role.

Fastbaud

The AccelePort *Xr* 920 PCI board supports speeds of up to 921600 baud. In order to be able to use these speeds under UNIX, the board offers a fastbaud function, which is enabled with the `fast_baud` command. If fastbaud is active, the speeds used at the serial interface are different to those set with the termio structure (e.g. via `stty`). The following values apply:

Specified baud rate	Actual baud rate used
50	57600
75	76800
110	115200
134	14400
150	57600
200	230400
300	76800
600	115200
1200	230400
1800	28800
2400	460800
4800	921600
9600	9600
19200	19200
38400	38400

DIAGNOSTICS

The `report` command with the specified board number provides information on the status of the board's ports. It is only if the board has the status RUNNING that the infos for the port are valid. The variables have the following meaning:

`port` Port number.

`mode` "`OFF`": Port is not in use, "`LOCAL DEVICE`": Port is being used as a local device without modem control, "`MODEM DEVICE`": Port is being used as a device with modem control.

`baud` "`normal`" for normal (1:1) baud rate mapping, "`fast`" if the fastbaud function is enabled.

`carrier`
"`CD`": CD signal is used as a carrier, "`DSR`": DSR signal is used as a carrier.

`m_signals`

The active modem signals (both input and output signals are displayed).

`t_q` Number of characters in the board's transmit queue.

`r_q` Number of characters in the board's receive queue.

In addition, trace data can be read from `/dev/digi_trace`.

The `digi` driver works under the MTT driver. This means that the (generic) trace options of the MTT are also available for the `digi` driver. The `digi` driver outputs its `strace(1M)` data under the ID 920.

NOTES

The `reset` command should not be used.

FILES

```
/dev/digi_adm
/dev/digi_trace
/etc/rc2.d/S10digi
/opt/digi/load/xrbios.bin
/opt/digi/load/xrfep.bin
```

SEE ALSO

`stty(1)`, `termio(7)`.

System Administrator's Reference Manual

display(7)
RM400 only

NAME

`display` - system console display

DESCRIPTION

The system console (and user's terminal) is composed of two separate pieces: the keyboard [see `keyboard(7)`] and the display. Because of their complexity, they are discussed in separate manual entries.

The display normally consists of 25 lines of 80 columns each. The effect of writing characters to the console or to one of its virtual screens (`/dev/console` or `/dev/vt`*xx*) depends on the characters. All characters written to `/dev/console` are first processed by the terminal interface [see `termio(7)`]. For example, mapping newline characters to carriage return plus newline, and expanding tabs to spaces, will be done before the following processing:

x Where *x* is not one of the following, displays *x*.

`BEL` Generates a bell (audible tone, no modulation).

`CR` Places the cursor at column 1 of the current line.

`LF`, `VT`
Places the cursor at the same column of the next line (scrolls if the the current line is line 25).

`FF` Clears the screen and places the cursor at line 1, column 1.

`S` Depends on the previous character: if a _ (underscore), see below; otherwise, if the cursor is not at column 1, it is moved to the left one position on the same line. If the cursor is at column 1 but not line 1, it is moved to column 79 of the previous line. Finally, if the cursor is at column 1, line 1, it is not moved.

`_S`*x* Sets the "underscore" attribute for the character *x* to be displayed.

`ESC`*x* Where *x* is *any* of the 256 possible codes (except for `c` and `[`), displays that value uninterpreted. This is useful for utilizing the full set of graphics available on the display. Note again that the characters are processed through the terminal interface prior to this escape sequence. Therefore, to get some of the possible 256 characters it is necessary that the character not be postprocessed. The easiest way to accomplish this is to turn off `OPOST` in the `c_oflag` field [see `termio(7)`]; however, this may have other side effects.

The display can be controlled by means of ANSI X3.64 escape sequences, which are specific sequences of characters, preceded by the ASCII character ESC. The escape sequences are the following:

`ESCc` Clears the screen and places the cursor at line 1, column 1.

`ESCQ` *n* '*string*'
Defines the function key *n* with *string*. The string delimiter ' may be any character not in *string*. Function keys are numbered 0 through 11 (F1 = 0, F2 = 1, etc.)

`ESC[` *n* `@`
Insert character – inserts *n* blanks at the current cursor position.

`ESC[` *n* `` ` ``
Horizontal position absolute – moves active position to column given by *n*.

`ESC[` *n* `A`
Cursor up – moves the cursor up *n* lines (default: *n*=1).

`ESC[` *n* `a`
Horizontal position relative – moves active position *n* characters to the right (default: *n*=1).

`ESC[` *n* `B`
Cursor down – moves the cursor down *n* lines (default: *n*=1).

ESC[*n* C
Cursor right – moves the cursor right *n* columns (default: *n*=1).

ESC[*n* c
Where n is 0 (underline cursor), 1 (blockcursor), or 2 (no cursor). 0 is the default value for *n*.

ESC[*n* D
Cursor left – moves the cursor left *n* columns (default: *n*=1).

ESC[*n* d
Vertical position absolute – moves active position to line given by *n*.

ESC[*n* E
Cursor next line – moves the cursor to column 1 of the next line, then down *n*–1 lines (default: *n*=1).

ESC[*n* e
Vertical position relative – moves the active position down *n* lines (default: *n*=1).

ESC[*n* F
Cursor previous line – moves the cursor to column 1 of the current line, then up *n* lines (default: *n*=1).

ESC[*n* G
Cursor horizontal position – moves the cursor to column *n* of the current line (default: *n*=1).

ESC[*n*; *m* H
Position cursor – moves the cursor to column *m* of line *n* (default: *n*=1, *m*=1).

ESC[*n*; *m* f
Position cursor – moves the cursor to column *m* of line *n* (default: *n*=1, *m*=1).

ESC[*n* J
Erase window – erases from the current cursor position to the end of the window if *n*=0, from the beginning of the window to the current cursor position if *n*=1, and the entire window if *n*=2 (default: *n*=0).

ESC[*n* K
Erase line – erases from the current cursor position to the end of the line if *n*=0, from the beginning of the line to the current cursor position if *n*=1, and the entire line if *n*=2 (default: *n*=0).

ESC[*n* L
Insert line – inserts *n* lines at the current cursor position (default: *n*=1).

ESC[*n* M
Delete line – deletes *n* lines starting at the current cursor position (default: *n*=1).

ESC[*n* P
Delete character – deletes *n* characters from a line starting at the current cursor position (default: *n*=1).

ESC[*n* S
Scroll up – scrolls the characters in the current window up *n* lines. The bottom *n* lines are cleared to blanks (default: *n*=1).

ESC[*n* T
Scroll down – scrolls the characters in the current window down *n* lines. The top *n* lines are cleared to blanks (default: *n*=1).

ESC[*n* X
Erase character – erases *n* character positions starting at the current cursor position (default: *n*=1).

ESC[*n* Z
Cursor Backward Tabulation – moves active position back *n* tab stops.

ESC[2h
Locks the keyboard and ignores keyboard input until unlocked. Characters are not saved.

ESC[2i
Sends the screen to the host. The current screen display is sent to the application.

ESC[2l

Unlocks the keyboard. Re-enables keyboard input.

ESC[*Ps* ; *Ps* ; m
Character attributes – each *Ps* is one of the following characters; multiple characters are separated by semicolons. These parameters apply to successive characters being displayed, in an additive manner (e.g. both bold and underscoring can be selected). (Default: *Ps*=0.)

Ps	Meaning
0	all attributes off (normal display)
	(white foreground, black background)
1	bold intensity
4	underscore on
7	reverse video
8	sets blank (non-display)
10	selects the primary font
11	selects the first alternate font; lets ASCII characters less than 32 be displayed as ROM characters (not implemented).
12	selects a second alternate font; sets the high bit of extended ASCII code before displaying as ROM characters.

ESC[*Pn* d
Moves the cursor to the line specified by *Pn*.

ioctl Calls

The following `ioctl` calls may be used to change the display used for the video monitor. If the virtual terminal has not been put in process mode (see the `VT_SETMODE ioctl`), setting the display mode to a non-text mode will turn off VT switching. VT switches will be re-enabled after the display mode has been reset to a text mode.

Note: All of the following `ioctls` are performed on either a file descriptor to the virtual terminals or to the special file `/dev/video`. `ioctls` to `/dev/video` are indicated with an asterisk (`*`). For the `ioctls` to `/dev/video` to work, the controlling `tty` for the process must be the virtual terminal on which the operation is to be performed. If the `tty` is not a virtual terminal, the return value will be –1 and `errno` will be set to `EINVAL`.

Switching to an invalid display mode for a display device will result in an error.

KIOCSOUND
Start sound generation. Turn on sound. The "arg" is the frequency desired. A frequency of 0 turns off the sound.

KDGETLED
Get keyboard LED status. The argument is a pointer to a character. The character will be filled with a boolean combination of the following values:

```
LED_SCR      0x01   (flag bit for scroll lock)
LED_CAP      0x04   (flag bit for caps lock)
LED_NUM      0x02   (flag bit for num lock)
```

KDSETLED
Set keyboard LED status. The argument is a character whose value is the boolean combination of the values listed under `KDGETLED`.

KDMKTONE
Generate a fixed length tone. The argument is a 32 bit value, with the lower 16 bits set to the frequency and the upper 16 bits set to the duration (in milliseconds).

KDGKBTYPE
: Get keyboard type. The argument is a pointer to a character type. The character will be returned with one of the following values:

```
KB_84         0x01   (84 key keyboard)
KB_101        0x02   (101 key keyboard)
KB_OTHER      0x03   (other type keyboard)
```

KDSETMODE (*)
: Set console in text or graphics mode. The argument is of type integer, which should contain one of the following values:

```
KD_TEXT       0x00   (sets console to text mode)
TKD_GRAPHICS 0x01   (sets console in graphics mode)
```

If the mode is set to KD_GRAPHICS and the Virtual Terminal is not in process mode (see the VT_SETMODE ioctl), no virtual terminal switches will be possible until the mode is reset to KD_TEXT, KD_TEXT0, or KD_TEXT1.

Note, the user is responsible for programming the color/graphics adaptor registers for the appropriate graphical state.

KDGETMODE
: Get current mode of console. Returns integer argument containing either KD_TEXT or KD_GRAPHICS as defined in the KDSETMODE ioctl description.

VT_OPENQRY
: Find an available virtual terminal. The argument is a pointer to a long. The long will be filled with the number of the first available VT that no other process has open or −1 if none are available.

VT_GETMODE (*)
: Determine what mode the active virtual terminal is currently in, either VT_AUTO or VT_PROCESS. The argument to the ioctl is the address of the following type of structure:

```
struct vt_mode
{
    char  mode;    /* VT mode */
    char  waitv;   /* if set, hang on writes when not active */
    short relsig;  /* signal to use for release request */
    short acqsig;  /* signal to use for display acquired */
    short frsig;   /* not used set to 0 */
};
#define VT_AUTO    0x00  /* automatic VT switching */
#define VT_PROCESS 0x01  /* process controls switching */
```

The vt_mode structure will be filled in with the current value for each field.

VT_GETSTATE (*)
: The VT_GETSTATE ioctl returns global virtual terminal state information. It returns the active virtual terminal in the v_active field, and the number of active virtual terminals and a bit mask of the global state in the vt_state field, where bit x is the state of vt x (1 indicates that the virtual terminal is open).

VT_SETMODE (*)
: Set the virtual terminal mode. The argument is a pointer to a vt_mode structure, as defined above.

VT_SENDSIG (*)
: The VT_SENDSIG ioctl specifies a signal (in vt_signal) to be sent to a bit mask of virtual terminals (in vt_state).

The data structure used by the VT_GETSTATE and VT_SENDSIG ioctls is:

```
struct vt_stat
{
    ushort v_active;  /* active vt */
```

```
        ushort v_signal;  /* signal to send (VT_SENDSIG) */
        ushort v_state;   /* vt bit mask (VT_SENDSIG and
                             VT_GETSTATE) */
    };
```

and is defined in `/usr/include/sys/vt.h`.

`VT_RELDISP` (*)
: Used to tell the virtual terminal manager that the display has or has not been released by the process. A non-zero argument indicates that the display has been released; a zero argument indicates refusal to release the display.

`VT_ACTIVATE`
: Makes the virtual terminal number specified in the argument the active VT. The VT manager will cause a switch to occur in the same manner as if a hotkey sequence had been typed at the keyboard. If the specified VT is not open or does not exist, the call will fail and errno will be set to `ENXIO`.

`KIOCINFO`
: This call tells the user what the device is.

`GIO_SCRNMAP` (*)
: This call gets the screen mapping table from the kernel.

`GIO_ATTR`
: This call returns the current screen attribute. The bits are interpreted as follows:

  Bit 0 determines underlining for black and white monitors (1 = underlining on).

  Bits 0-2, for color monitors only, select the foreground color. The following list indicates what colors are selected by the given value:

```
The value 0 selects black.
The value 1 selects red.
The value 2 selects green.
The value 4 selects blue.
The value 5 selects magenta.
The value 6 selects cyan.
The value 7 selects white.
```

  Bit 3 is the intensity bit (1 = intensity on).

  Bits 4-6, for color monitors only, select the background color. For a list of colors and their values, see the list under foreground colors.

  Bit 7 is the blink bit (1 = blink on).

`GIO_COLOR` (*)
: This call returns a non-zero value if the current display is a color display, otherwise, it returns a zero.

`PIO_SCRNMAP`
: This call puts the screen mapping table in the kernel.

  The screen mapping table maps extended ASCII (8-bit) characters to ROM characters. It is an array [256] of `char` (`typedef scrnmap_t`) and is indexed by extended ASCII values. The value of the elements of the array are the ROM character to display.

  For example, the following will change the ASCII character "#" to be displayed as an British pound sign.

```
#include <sys/console.h>
change_pound()
{
    scrnmap_t scrntab;
    /* get screen mapping table of standard output */
    if (ioctl(0,GIO_SCRNMAP, scrntab)==-1)
    {
        perror("screenmap read");
```

```
            exit(-1);
        }
        /* 156 is the ROM value of British pound sign and 30
           is the ASCII value of '#'. */
        scrntab[30] = 156;
        if (ioctl(0, PIO_SCRNMAP, scrntab) == -1)
        {
            perror("screenmap write");
            exit(-1);
        }
    }
```

NOTES

Although it is possible to write character sequences which set arbitrary bits on the screen in any of the three graphics modes, this mode of operation is not currently supported.

FILES

`/dev/console`

`/dev/vt00-`*n*

`/dev/video`

`/usr/include/sys/kd.h`

SEE ALSO

`stty`(1), `ioctl`(2), `console`(7), `keyboard`(7), `termio`(7).

System Administrator's Reference Manual

eip2(7)
RM600 only

NAME

`eip2 -` Environment Interface Processor 2

DESCRIPTION

The EIP2 processor is used to switch on and off the system, and control the fans, temperature, and power voltage. The operating panel of a system cabinet, i.e. LCD (Liquid Crystal Display), LED (Light Emitting Diode), menu button, etc. is also addressed via the EIP2 processor. Internal or external BBUs (Battery Backup Unit) are also controlled via the EIP2. One EIP2 is available per system cabinet. The EIPs2 of the individual cabinets communicate with one another via the CAN bus. The EIP2 of the central cabinet is connected to the CSI (Central, Services & Interface Processor) and is called the master EIP2, whereas the EIPs2 of the other cabinets are called slave EIP2.

The master EIP2 can be addressed via the CSI maintenance function [see `csimaint(7)`]. Commands are either processed directly by the master or passed transparently to the slave EIP2 depending on the address specified in the command. The control panel of the central cabinet differs from that of an extension cabinet which means that a number of commands referring to them are restricted on the master EIP2.

The following functions can be executed via `/dev/csimaint` for every command called:

- Define EIP2 version(only master EIP2)
- Determine the status of a cabinet
- Describe the LCD(only master EIP2)
- Query menu button(only master EIP2)
- Change color of the LED
- Determine the EIP2 test status
- Determine the firmware release
- Query the BBU status
- Query the UPS status

FILES

`/dev/csimaint`

SEE ALSO

`csimaint(7)`.

System Administrator's Reference Manual

error(7)

NAME

`error` - log device for system state and errors

SYNOPSIS

`/dev/error`

DESCRIPTION

`/dev/error` is the pseudo-device used to pass kernel log entries to the user level error daemon, `errord`. The error daemon processes and saves the log entries.

EXAMPLE

```
mknod /dev/error c 206 0
```

SEE ALSO

`errord`(8).

System Administrator's Reference Manual

fd(7)
RM400 only

NAME

`fd` - floppy disk driver

DESCRIPTION

Floppy disks are used on the RM400 mainly for access to *key disks* during the installation of optional software packages.

Floppy disks are accessed through the special files in the `/dev/at/flp` directory (see **FILES**).

The following devices are supported:

`[r]fx3d[t]`
: These devices refer to *double density* floppy disks with 9 sectors per track, the devices ending in `t` access the entire disk, providing 720 kbytes of formatted storage, the devices without the `t` access all but the first cylinder, providing 711 kbytes of formatted storage.

`[r]fx3h[t]`
: These devices refer to *high density* floppy disks with 18 sectors per track, the devices ending in `t` access the entire disk, providing 1.44 mbytes of formatted storage, the devices without the `t` access all but the first cylinder, providing 1.422 mbytes of formatted storage.

`[r]f0[t]`
: These are the *auto-detection* devices that probe the floppy disk inside the drive in order to determine the format. Again, the devices ending in `t` access the entire disk, providing 720 kbytes or 1.44 mbytes of formatted storage, those without the `t` access all but the first cylinder, providing 711 kbytes or 1.422 mbytes of formatted storage. Obviously these devices can only be used if the floppy disk inside the drive has already been formatted.

All `t`-devices have 80 cylinders, 2 heads and 512 bytes per sector.

DEVICE NUMBERS

Major Device Number

The floppy disk driver `fd` has the major number 23 for both the block special and character special devices.

Minor Device Number

The minor device number selects one of the recording formats and the partition of the floppy disk the user wishes to access.

The recording formats currently supported are:

`6`: for *double density* floppy disks (`[r]fx3d[t]`)

`7`: for *high density* floppy disks (`[r]fx3h[t]`)

`8`: for the *auto-detection* mechanism (`[r]f0[t]`)

The partitions currently supported are:

`0`: for access to the entire disk (all 80 cylinders, device name ending in `t`)

`2`: for access to all but the first cylinder of the disk (i.e. to the cylinders 1 to 79, device names not ending in `t`)

The minor number can then be computed by "*format* * 16 + *partition* * 2".

IOCTLs

The `ioctl(2)` system calls have the form

```
ioctl(fd, command, arg)
int fd, command;
```

```
/* see below for arg */
```

The floppy disk driver `fd` currently supports 5 `ioctl`-requests:

`V_GETPARMS`

Retrieve device parameters (device type, number of heads, number of cylinders, number of sectors per track, etc.). *arg* is a pointer to the following structure:

```
struct disk_parms
{
    char    dp_type;       /* Disk type (see below) */
    unchar  dp_heads;      /* Number of heads */
    ushort  dp_cyls;       /* Number of cylinders */
    ushort  dp_sectors;    /* Number of sectors/track */
    ushort  dp_secsiz;     /* Number of bytes/sector for this
                              partition */
    ushort  dp_ptag;       /* Partition tag */
    ushort  dp_pflag;      /* Partition flag */
    daddr_t dp_pstartsec;  /* Starting absolute sector number */
    daddr_t dp_pnumsec;    /* Number of sectors */
}
```

The floppy disk driver will always set `dp_type` to DPT_FLOPPY to denote a floppy disk drive. Currently the driver will always set `dp_heads` to 2, `fd_cyls` to 80, `dp_secsiz` to either 9 (*double density*) or 18 (*high density*), and `dp_secsiz` to 512. `dp_ptag` and `dp_pflag` are currently unused and set to 0. `dp_startsec` and `dp_numsec` are set according to the selected partition.

The total capacity of the partition can be calculated with the formula: "`arg->dp_secsiz * arg->dp_pnumsec`".

`V_FORMAT`

Format one or more tracks with a specified *interleave factor* (this function is not available on the auto-detection devices `f0`, `rf0`, `f0t` and `rf0t`). *arg* is a pointer to the following structure:

```
struct ia_fmt
{
    ushort start_trk;  /* first track # */
    ushort num_trks;   /* number of tracks to format */
    ushort intlv;      /* interleave factor */
};
```

For compatibility with industry standard PCs, the sectors in each track will be numbered starting with sector number 1.

`F_DOR`

Reset the floppy disk drive to a known state, *arg* is a pointer to a character. If that character has the third-lowest bit reset, then the drive will be reset.

`F_FCR`

Change the data transfer rate, *arg* is a pointer to a character. The two least significant bits select one of three transfer rates:

`0` selects 500 kbits per second transfer rate,

`1` selects 300 kbits per second transfer rate,

`2` selects 250 kbits per second transfer rate, and

`3` selects 125 kbits per second transfer rate (this value is currently unsupported).

`F_DTYP`

Determine the drive type.

arg is a pointer to an unsigned character, which will be set to 4 denoting a 3.5" floppy disk drive.

FILES

```
/dev/at/flp/f0
/dev/at/flp/rf0
/dev/at/flp/f0t
/dev/at/flp/rf0t
/dev/at/flp/fx3d
/dev/at/flp/rfx3d
/dev/at/flp/fx3dt
/dev/at/flp/rfx3dt
/dev/at/flp/fx3h
/dev/at/flp/rfx3h
/dev/at/flp/fx3ht
/dev/at/flp/rfx3ht
```

System Administrator's Reference Manual

filesystem(7)

NAME

`filesystem` - file system organization

SYNOPSIS

/
/usr

DESCRIPTION

The Reliant UNIX file system tree is organized for administrative convenience. Distinct areas within the file system tree are provided for files that are private to one machine, files that can be shared by multiple machines of a common architecture, files that can be shared by all machines, and home directories. This organization allows sharable files to be stored on one machine but accessed by many machines using a remote file access mechanism such as NFS. Grouping together similar files makes the file system tree easier to upgrade and manage.

The file system tree consists of a root file system and a collection of mountable file systems. The `mount(2)` program attaches mountable file systems to the file system tree at mount points (directory entries) in the root file system or other previously mounted file systems. Two file systems, `/` (the root) and `/usr`, must be mounted in order to have a completely functional system. The root file system is mounted automatically by the kernel at boot time; the `/usr` file system is mounted by the `/etc/rc.boot` script, which is run as part of the booting process.

The root file system contains files that are unique to each machine. It contains the following directories:

`/dev` Character and block special files. These device files provide hooks into hardware devices or operating system facilities. Typically, device files are built to match the kernel and hardware configuration of the machine.

`/dev/term`
Terminal devices.

`/dev/pts`
Pseudo-terminal devices.

`/dev/xt`
Devices used by `layers`.

`/dev/sxt`
Shell layers device files used by `shl`.

`/etc` Machine-specific administrative configuration files and system administration databases. `/etc` may be viewed as the home directory of a machine, the directory that in a sense defines the machine's identity. Executable programs are no longer kept in `/etc`.

`/home` Root of a subtree for user directories.

`/mnt` Temporary mount point for file systems. This is an empty directory on which file systems may be temporarily mounted.

`/opt` Root of a subtree for add-on application packages.

`/proc` Root of a subtree for the process file system.

`/sbin` Essential executables used in the booting process and in manual system recovery. The full complement of utilities is available only after `/usr` is mounted.

`/tmp` Temporary files; initialized to empty during the boot operation.

`/var` Root of a subtree for varying files. Varying files are files that are unique to a machine but that can grow to an arbitrary (that is, variable) size. An example is a log file.

`/var/adm`
System logging and accounting files.

`/var/cron`
: `cron`'s log file.

`/var/mail`
: Where users' mail is kept.

`/var/opt`
: Top-level directory used by application packages.

`/var/preserve`
: Backup files for `vi(1)` and `ex(1)`.

`/var/spool`
: Subdirectories for files used in printer spooling, mail delivery, `cron(1M)`, `at(1)`, etc.

`/var/tmp`
: Transitory files; initialized to empty during the boot operation.

Because it is desirable to keep the root file system small and not volatile, on disk-based systems larger file systems are often mounted on `/home`, `/opt`, `/usr`, and `/var`.

The file system mounted on `/usr` contains architecture-dependent and architecture-independent sharable files. The subtree rooted at `/usr/share` contains architecture-independent sharable files; the rest of the `/usr` tree contains architecture-dependent files. By mounting a common remote file system, a group of machines with a common architecture may share a single `/usr` file system. A single `/usr/share` file system can be shared by machines of any architecture. A machine acting as a file server may export many different `/usr` file systems to support several different architectures and operating system releases. Clients usually mount `/usr` read-only so that they do not accidentally change any shared files. The `/usr` file system contains the following subdirectories:

`/usr/bin`
: Most system utilities.

`/usr/sbin`
: Executables for system administration.

`/usr/games`
: Game binaries and data.

`/usr/include`
: Include header files (for C programs, etc).

`/usr/lib`
: Program libraries, various architecture-dependent databases, and executables not invoked directly by the user (system daemons, etc).

`/usr/share`
: Subtree for architecture-independent sharable files.

`/usr/share/man`
: Subdirectories for online reference manual pages (if present).

`/usr/share/lib`
: Architecture-independent databases.

`/usr/src`
: Source code for utilities and libraries.

`/usr/ucb`
: Berkeley compatibility package binaries.

`/usr/ucbinclude`
: Berkeley compatibility package header files.

`/usr/ucblib`
: Berkeley compatibility package libraries.

A machine with disks may export root file systems, swap files, and `/usr` file systems to diskless or partially-disked machines that mount them into the standard file system hierarchy. The standard directory tree for sharing these file systems from a server is:

`/export`
The default root of the exported file system tree.

`/export/exec/`*architecture-name*
The exported `/usr` file system supporting *architecture-name* for the current release.

`/export/exec/`*architecture-name*`.`*release-name*
The exported `/usr` file system supporting *architecture-name* for Reliant UNIX *release-name*.

`/export/exec/share`
The exported common `/usr/share` directory tree.

`/export/exec/share.`*release-name*
The exported common `/usr/share` directory tree for Reliant UNIX *release-name*.

`/export/root/`*hostname*
The exported root file system for *hostname*.

`/export/swap/`*hostname*
The exported swap file for *hostname*.

`/export/var/`*hostname*
The exported `/var` directory tree for *hostname*.

SEE ALSO

`at`(1), `ksh`(1), `sh`(1), `vi`(1), `init`(1M), `mknod`(1M), `mount`(1M), `fsck`(1M), `intro_sarm4`(4).

keyboard(7) - sxt(7)

System Administrator's Reference Manual

keyboard(7)
RM400 only

NAME

`keyboard` - system console keyboard

DESCRIPTION

The system console is composed of two separate pieces: the keyboard and the display [see `display`(7)].

The keyboard is used to type data and send certain control signals to the computer. Reliant UNIX software performs terminal emulation on the console screen and keyboard, and, in doing so, makes use of several particular keys and key combinations. These keys and key combinations have special names that are unique to the Reliant UNIX system, and may or may not correspond to the keytop labels on your keyboard. When you press a key, one of the following happens:

- An ASCII value is entered.
- The meaning of another key, or keys, is changed.
- A string is sent to the computer.
- A function is initiated.

When a key is pressed (a keystroke), the keyboard sends a scancode to the computer. This scancode is interpreted by the keyboard driver. The actual code sequence delivered to the terminal input routine [see `termio`(7)] is defined by a set of internal tables in the driver. These tables can be modified by software (see the discussion of `ioctl` calls below). In addition, the driver can be instructed not to do translations, delivering the keyboard up/down scan codes directly.

Changing Meanings

The action performed by a key can be changed by using certain keys in combination. For example, the `SHIFT` key changes the ASCII values of the alphanumeric keys. Holding down the `CTRL` key while pressing another key sends a control code (such as `CTRL-D`, `CTRL-S`, and `CTRL-Q`). Holding down the `ALT` key also modifies a

key's value. The SHIFT, CTRL, and ALT keys can be used in combination.

Switching Screens

To change screens (virtual terminals), first run the vtlmgr command [see vtlmgr(1M)]. Switch the current screen by typing ALT-SYSREQ (also labeled ALT-PRINTSCRN on some systems) followed by a key which identifies the desired screen. Any active screen may be selected by following ALT-SYSREQ with F*n*, where F*n* is one of the function keys. F1 refers to the first virtual terminal screen, F2 refers to the second virtual terminal screen, etc. ALT-SYSREQ "h" refers to the main console display (/dev/console). The next active screen can be selected with ALT-SYSREQ "n", and the previous screen can be selected with ALT-SYSREQ "p".

The default screen switch enable sequence (ALT-SYSREQ) is configurable. The SYSREQ table entry can be modified by software (see discussion of ioctl calls below).

Special Keys

The following table shows which keys on a typical console correspond to Reliant UNIX system keys. In this table, a hyphen (-) between keys means you must hold down the first key while pressing the second. The mapping between characters which generate signals and the signal actually generated is set with stty(1) , and may be changed [see stty(1)].

Name	Keytop	
INTR	DEL	Stops current action and returns to th key.
BACKSPACE	←	Deletes the first character to the left (arrow (←) on its keytop, but you can
CTRL-D	CTRL-D	Signals the end of input from the key
CTRL-H	CTRL-H	Deletes the first character to the left (
CTRL-Q	CTRL-Q	Restarts printing after it has been sto
CTRL-S	CTRL-S	Suspends printing on the screen (do
CTRL-U	CTRL-U	Deletes all characters on the current
CTRL-\	CTRL-\	Quits current command and creates
ESCAPE	ESC	Special code for some programs. Fo the vi(1) text editor.
RETURN	or ENTER	Terminates a command line and initi

Name	Keytop	
F*n*	F*n*	Function key *n*. F1-F12 are unshifte through F12, and F37-F48 are CTRI The next F*n* keys (F49-F60) are on □ F49-7 F55-6 □ F50-8 F56-+ □ F51-9 F57-1 □ F52-– □F58-2 □ F53-4 F59-3 □ F54-5 F60-0

Keyboard Map

The keyboard mapping structure is defined in /usr/include/sys/kd.h. Each key can have ten states. The first eight states are:

- BASE
- CTRL-SHIFT
- SHIFT
- ALT-SHIFT
- CTRL
- ALT-CTRL
- ALT
- ALT-CTRL-SHIFT

The two remaining states are indicated by two special bytes. The first byte is a "special state" byte whose bits indicate whether the key is "special" in one or more of the first eight states. The second byte is one of four codes represented by the characters C, N, B, or O which indicate how the lock keys affect the particular key.

The following table describes the default keyboard mapping. All values, except for special keywords (which are described later), are ASCII character values.

Heading
Description

SCAN CODE
This column contains the scan code generated by the keyboard hardware when a key is pressed. There are no table entries for the scan code generated by releasing a key.

BASE This column contains the normal value of a key press.

SHIFT
This column contains the value of a key press when the SHIFT is also being held down.

LOCK This column indicates which lock keys affect that particular key:

- C indicates CAPSLOCK
- N indicates NUMLOCK
- B indicates both
- O indicates "none"

The remaining columns are the values of key presses when combinations of the CTRL, ALT and SHIFT keys are also held down.

The SRQTAB column entry is included in this table to provide a simple index of the default virtual terminal key selectors to the scan code to which it is assigned. The actual SRQTAB table is a stand-alone table which can be read or written via the KDGKBENT and KDSKBENT ioctl calls.

SCAN CODE	BASE	SHIFT	CTRL	CTRL SHIFT
0	nop	nop	nop	nop
1	esc	esc	esc	esc
2	'1'	'!'	'1'	'1'
3	'2'	'@'	nul	nul
4	'3'	'#'	'3'	'3'
5	'4'	'$'	'4'	'4'
6	'5'	'%'	'5'	'5'
7	'6'	'^'	rs	rs
8	'7'	'&'	'7'	'7'

9	'8'	'*'	'8'	'8'
10	'9'	'('	'9'	'9'
11	'0'	')'	'0'	'0'
12	'-'	'_'	ns	ns
13	'='	'+'	'='	'+'
14	bs	bs	bs	bs
15	ht	gs	ht	gs
16	'q'	'Q'	dc1	dc1
17	'w'	'W'	etb	etb
18	'e'	'E'	enq	enq
19	'r'	'R'	dc2	dc2
20	't'	'T'	dc4	dc4
21	'y'	'Y'	em	em
22	'u'	'U'	nak	nak
23	'i'	'I'	ht	ht

SCAN CODE	BASE	SHIFT	CTRL	CTRL SHIFT
24	'o'	'O'	si	si
25	'p'	'P'	dle	dle
26	'['	'{'	esc	esc
27	']'	'}'	gs	gs
28	cr	cr	cr	cr
29	lctrl	lctrl	lctrl	lctrl
30	'a'	'A'	soh	soh
31	's'	'S'	dc3	dc3
32	'd'	'D'	eot	eot
33	'f'	'F'	ack	ack
34	'g'	'G'	bel	bel
35	'h'	'H'	bs	bs
36	'j'	'J'	nl	nl
37	'k'	'K'	vt	vt
38	'l'	'L'	np	np
39	';'	':'	';'	':'
40	'\''	'"'	'\''	'"'
41	'`'	'~'	'`'	'~'
42	lshift	lshift	lshift	lshift

43	'\\'	'|'	fs	fs
44	'z'	'Z'	sub	sub
45	'x'	'X'	can	can
46	'c'	'C'	etx	etx
47	'v'	'V'	syn	syn
48	'b'	'B'	stx	stx
49	'n'	'N'	so	so
50	'm'	'M'	cr	cr
51	','	'<'	','	'<'
52	'.'	'>'	'.'	'>'
53	'/'	'?'	'/'	'?'
54	rshift	rshift	rshift	rshift
55	'*'	'*'	'*'	'*'
56	lalt	lalt	lalt	lalt
57	' '	' '	nul	nul
58	clock	clock	clock	clock
59	fkey00	fkey12	fkey24	fkey36
60	fkey01	fkey13	fkey25	fkey37
61	fkey02	fkey14	fkey26	fkey38
62	fkey03	fkey15	fkey27	fkey39
63	fkey04	fkey16	fkey28	fkey40

SCAN CODE	BASE	SHIFT	CTRL	CTRL SHIFT
64	fkey05	fkey17	fkey29	fkey41
65	fkey06	fkey18	fkey30	fkey42
66	fkey07	fkey19	fkey31	fkey43
67	fkey08	fkey20	fkey32	fkey44
68	fkey09	fkey21	fkey33	fkey45
69	nlock	nlock	nlock	nlock
70	ack	ack	'|'	'|'
71	fkey48	'7'	fkey48	'7'
72	fkey49	'8'	fkey49	'8'
73	fkey50	'9'	fkey50	'9'
74	fkey51	'-'	fkey51	'-'
75	fkey52	'4'	fkey52	'4'
76	fkey53	'5'	fkey53	'5'

77	fkey54	'6'	fkey54	'6'
78	fkey55	'+'	fkey55	'+'
79	fkey56	'1'	fkey56	'1'
80	fkey57	'2'	fkey57	'2'
81	fkey58	'3'	fkey58	'3'
82	fkey59	'0'	fkey59	'0'
83	del	'.'	del	'.'
84	fkey59	fkey25	fkey59	nop
85	fkey57	fkey57	fkey57	fkey57
86	nop	nop	nop	nop
87	fkey10	fkey22	fkey34	fkey46
88	fkey11	fkey23	fkey35	fkey47
89	nop	nop	nop	nop
90	nop	nop	nop	nop
91	nop	nop	nop	nop
92	nop	nop	nop	nop
93	nop	nop	nop	nop
94	nop	nop	nop	nop
95	nop	nop	nop	nop
96	nop	nop	nop	nop
97	nop	nop	nop	nop
98	nop	nop	nop	nop
99	nop	nop	nop	nop
100	nop	nop	nop	nop
101	nop	nop	nop	nop
102	nop	nop	nop	nop
103	nop	nop	nop	nop

SCAN CODE	BASE	SHIFT	CTRL	CTRL SHIFT
104	nop	nop	nop	nop
105	nop	nop	nop	nop
106	nop	nop	nop	nop
107	fkey52	fkey52	fkey52	fkey52
108	nop	nop	nop	nop
109	nop	nop	nop	nop
110	nop	nop	nop	nop

111	fkey50	fkey50	nop	nop
112	nop	nop	nop	nop
113	nop	nop	nop	nop
114	agr	agr	agr	agr
115	rctrl	rctrl	rctrl	rctrl
116	cr	cr	cr	cr
117	'/'	'/'	nop	nop
118	nop	nop	nop	nop
119	brk	brk	brk	brk
120	fkey49	fkey49	nop	nop
121	del	del	del	del
122	fkey56	fkey56	nop	nop
123	fkey59	fkey59	nop	nop
124	nop	nop	nop	nop
125	fkey54	fkey54	nop	nop
126	fkey58	fkey58	nop	nop
127	fkey48	fkey48	nop	nop

The following table lists the value of each of the special keywords used in the preceding tables. The keywords are only used in the preceding tables for readability. In the actual keyboard map, a special keyword is represented by its value with the corresponding "special state" bit being set.

Name:Value:Meaning
nop:0:No operation – no action from keypress
lshift:2:Left-hand shift
rshift:3:Right-hand shift
clock:4:Caps lock
nlock:5:Numeric lock
slock:6:Scroll lock
alt:7:Alt key
btab:8:Back tab key – generates fixed sequence (ESC[Z)
ctrl:9:Control key
lalt:10:Left-han

d alt key

ralt:11:right-ha
nd alt key

lctrl:12:Left-ha
nd control key

rctrl:13:Right-h
and control key

agr:14:ALT-GR
key (European
keyboards only)

fkey1:27:Functio
n key #1

.::.

.::.

.::.

fkey96:122:Funct
ion key #96

sysreq:123:Syste
m request

brk:124:Break
key

escn:125:Generat
e an ESC N *x*
sequence, where *x* is
the un-alt'ed value of
the scan code

esco:126:Generat
e an ESC O *x*
sequence, where *x* is
the un-alt'ed value of
the scan code

escl:127:Generat
e an ESC L *x*
sequence, where *x* is
the un-alt'ed value of
the scan code

rboot:128:Reboot
system

debug:129:Invoke
kernel debugger

NEXT:130:Switch
to next virtual
terminal on
queue

PREV:131:Switch
to previous
virtual terminal
on queue

FNEXT:132:Forced

switch to next virtual terminal on queue
FPREV:133:Forced switch to previous virtual terminal on queue

Name:Value:Meaning
VTF:134:Virtual Terminal First (VT00)
.::.
.::.
.::.
VTL:148:Virtual Terminal Last (VT14)
MGRF:149:Virtual Terminal Manager First. Allows assigning special significance to key sequence for actions by virtual terminal layer manager. Used in SRQTAB table.
.::.
.::.
.::.
MGRL:179:Virtual Terminal Manager Last. Used in SRQTAB table.

The following table lists names and decimal values for ASCII characters in the preceding table. Names are used in place of numeric constants to make it easier to read the scan code table. Only the decimal values are placed in the `ioctl` buffer. These values are taken from `ascii(5)` .

Name:Value:Name:Value:Name:Value:Name:Value		
nul:0:ht:9:dc2:18:esc:27		
soh:1:nl:10:dc3:19:fs:28		
stx:2:vt:11:dc4:20:gs:29		

etx:3:np:12:nak: 21:rs:30 eot:4:cr:13:syn: 22:ns:31 enq:5:so:14:etb: 23:del:127 ack:6:si:15:can: 24 bel:7:dle:16:em: 25 bs:8:dc1:17:sub: 26		

String Key Mapping

The string mapping table is an array of 512 bytes (typedef `strmap_t`) containing null-terminated strings that redefine the function keys. The first null-terminated string is assigned to the first function key, the second string is assigned to the second function key, etc.

There is no limit to the length of any particular string as long as the whole table does not exceed 512 bytes, including nulls. To make a string a null, add extra null characters. The following table contains default function key values.

Default Function Key Values
Function::Shift:Ctrl:Ctrl Shift Key #:Function:Function:Function:Function
1:ESC OP:ESC Op:ESC OP:ESC Op 2:ESC OQ:ESC Oq:ESC OQ:ESC Oq 3:ESC OR:ESC Or:ESC OR:ESC Or 4:ESC OS:ESC Os:ESC OS:ESC Os 5:ESC OT:ESC Ot:ESC OT:ESC Ot 6:ESC OU:ESC Ou:ESC OU:ESC Ou 7:ESC OV:ESC Ov:ESC OV:ESC Ov 8:ESC OW:ESC Ow:ESC OW:ESC Ow 9:ESC OX:ESC Ox:ESC OX:ESC Ox 10:ESC OY:ESC

Oy:ESC OY:ESC Oy

11:ESC OZ:ESC
Oz:ESC OZ:ESC Oz

12:ESC OA:ESC
Oa:ESC OA:ESC Oa

ioctl Calls

KDGKBMODE

This call gets the current keyboard mode. It returns one of the following values, as defined in /usr/include/sys/kd.h:

```
#define   K_RAW     0x00   /* Send row scan codes */
#define   K_XLATE   0x01   /* Translate to ASCII */
```

KDSKBMODE

This call sets the keyboard mode. The argument to the call is either K_RAW or K_XLATE. By using raw mode, the program can see the raw up/down scan codes from the keyboard. In translate mode, the translation tables are used to generate the appropriate character code.

KDGKBTYTE

This call gets the keyboard type. It returns one of the following values, as defined in /usr/include/sys/kd.h:

```
#define   KB_84      0x00   /* 84 key keyboard */
#define   KB_101     0x01   /* 101 key keyboard */
#define   KB_OTHER   0x03   /* Other type keyboard */
```

KDGKBENT

This call reads one of the entries in the translation tables. The argument to the call is the address of one of the following structures, defined in /usr/include/sys/kd.h, with the first two fields filled in:

```
struct kbentry
{
    unchar kb_table;  /* Table to use */
    unchar kb_index;  /* Entry in table */
    ushort kb_value;  /* Value to get/set */
};
```

Valid values for the kb_table field are:

```
#define   K_NORMTAB      0x00   /* BASE*/
#define   K_SHIFTTAB     0x01   /* SHIFT*/
#define   K_ALTTAB       0x02   /* ALT */
#define   K_ALTSHIFTTAB  0x03   /* SHIFT+ALT */
#define   K_SRQTAB       0x04   /* Select SYSREQ table */
```

The ioctl will get the indicated entry from the indicated table and return it in the third field.

The K_SRQTAB value for the kb_table field allows access to the scancode indexed table which allows assignment of a given virtual terminal selector (K_VTF—K_VTL) or the virtual terminal layer manager (K_MGRF—K_MGRL) "specialkey" assignments.

The virtual terminal selector (K_VTF) is normally associated with /dev/vt00, on which the user login shell is commonly found. The following terminal selectors also are used to select virtual terminals:

```
K_VTF+1 for the 1st virtual terminal (/dev/vt01)
K_VTF+2 for the 2nd virtual terminal (/dev/vt02)
. . .
K_VTF+12 for the 12th virtual terminal (/dev/vt12)
```

KDSKBENT

This call sets an entry in one of the translation tables. It uses the same structure as the KDGKBENT ioctl, but with the third field filled in with the value that should be placed in the translation table. This can be used to partially or completely remap the keyboard.

The kd driver provides support for virtual terminals. The console minor device, /dev/vtmon, provides virtual terminal key requests from the kd driver to the process that has /dev/vtmon open. Two ioctls are provided for virtual terminal support:

VT_GETSTATE
: The VT_GETSTATE ioctl returns global virtual terminal state information. It returns the active virtual terminal in the v_active field, and the number of active virtual terminals and a bit mask of the global state in the vt_state field, where bit□x is the state of vt□x (1 indicates that the virtual terminal is open).

VT_SENDSIG
: The VT_SENDSIG ioctl specifies a signal (in vt_signal) to be sent to a bit mask of virtual terminals (in vt_state).

The data structure used by the VT_GETSTATE and VT_SENDSIG ioctls is:

```
struct vt_stat
{
  ushort v_active;  /* active vt */
  ushort v_signal;  /* signal to send (VT_SENDSIG) */
  ushort v_state;   /* vt bit mask (VT_SENDSIG + VT_GETSTATE) */
};
```

and is defined in /usr/include/sys/vt.h.

VT_OPENQRY
: The VT_OPENQRY ioctl is used to get the next available virtual terminal. This value is set in the last argument of the ioctl(2) call.

GIO_KEYMAP
: This call gets the entire keyboard mapping table from the kernel. The structure of the argument is given in /usr/include/sys/kd.h.

PIO_KEYMAP
: This call sets the entire keyboard mapping table. The structure of the argument is given in /usr/include/sys/kd.h.

GIO_STRMAP
: This call gets the string key mapping table from the kernel. The structure of the argument is given in /usr/include/sys/kd.h.

PIO_STRMAP
: This call sets the string key mapping table. The structure of the argument is given in /usr/include/sys/kd.h.

TIOCKBOF
: Extended character codes are disabled. This is the default mode.

TIOCKBON
: Allows extended characters to be transmitted to the user program. The extended characters are transmitted as a null byte followed by a second byte containing the character's extended code. When a true null byte is sent, it is transmitted as two consecutive null bytes.

When the keyboard is fully enabled, an 8-bit character code can be obtained by holding down the ALT key and entering the 3-digit decimal value of the character from the numeric keypad. The character is transmitted when the ALT key is released.

Some keyboard characters have special meaning. Under default operations, pressing the DELETE key generates an interrupt signal which is sent to all processes designated with the associated control terminal. When the keyboard is fully enabled, holding down the ALT key and pressing the 8 key on the home keyboard (not on the numeric keypad) returns a null byte followed by 0x7F. This will produce the same effect as the DELETE key (0x7F) unless you have executed the stty(1) command with the

`-isig` option.

`KBENABLED`
If the keyboard is fully enabled (`TIOCKBON`), a non-zero value will be returned. If the keyboard is not fully enabled (`TIOCKBOF`), a value of zero will be returned.

`GETFKEY`
Obtains the current definition of a function key. The argument to the call is the address of one of the following structures defined in `/usr/include/sys/kd.h`:

```
struct fkeyarg
{
    unsigned int keynum;
    char keydef [MAXFK];  /* Comes from ioctl.h via comcrt.h */
    char flen;
};
```

The function key number must be passed in `keynum` (see `arg` structure above). The string currently assigned to the key will be returned in `keydef` and the length of the string will be returned in `flen` when the `ioctl` is performed.

`SETFKEY`
Assigns a given string to a function key. It uses the same structure as the `GETFKEY ioctl`. The function key number must be passed in `keynum`, the string must be passed in `keydef`, and the length of the string (number of characters) must be passed in `flen`.

FILES

`/dev/console`
`/dev/vt00-`*n*
`/usr/include/sys/kd.h`

SEE ALSO

`stty(1)`, `vtlmgr(1M)`, `ioctl(2)`, `ascii(5)`, `console(7)`, `display(7)`, `termio(7)`.

System Administrator's Reference Manual

lar(7)

NAME

`lar` - Lockout Auto Recovery

DESCRIPTION

If an individual component in a system fails or malfunctions, it is important to ensure that this single component does not prevent operation of the entire system.

The LAR concept was developed with this in mind. This concept allows online monitoring of the following components:

– on RM600 type computers
boards, SCSI devices (disks, CD-ROM drives, tape drives), BBUs, power supplies and fan-out units as well as the ambient temperature.

– on RM200/300/400 type computers
main memory (RM400C only), SCSI devices (disks, CD-ROM drives), BBUs, power supplies and fan-out units as well as primary power failures and the ambient temperature.

For further information, please refer to the manual *Diagnostics and Troubleshooting*, chap. *LAR concept*.

FILES

`/etc/init.d/lar_rc`
`/sbin/scsilar_rc`
`/var/lar/hwconfig`

SEE ALSO

`act_dev`(1M), `deact_dev`(1M),
only RM400: `act_ctlr`(1M), `deact_ctlr`(1M),
only RM600: `reset_lar`(1M), `set_lar`(1M),
`olr`(7).

System Administrator's Reference Manual

ldterm(7)

NAME

`ldterm` - standard STREAMS terminal line discipline module

DESCRIPTION

`ldterm` is a STREAMS module that provides most of the `termio(7)` terminal interface. This module does not perform the low-level device control functions specified by flags in the `c_cflag` word of the `termio`/`termios` structure or by the `IGNBRK`, `IGNPAR`, `PARMRK`, or `INPCK` flags in the `c_iflag` word of the `termio`/`termios` structure; those functions must be performed by the driver or by modules pushed below the `ldterm` module. All other `termio`/`termios` functions are performed by `ldterm`; some of them, however, require the cooperation of the driver or modules pushed below `ldterm` and may not be performed in some cases. These include the `IXOFF` flag in the `c_iflag` word and the delays specified in the `c_oflag` word.

`ldterm` also handles EUC and multi-byte characters and `mapchan` tables.

The remainder of this section describes the processing of various STREAMS messages on the read and write side.

Read side behavior

Various types of STREAMS messages are processed as follows:

`M_BREAK`
: When this message is received, either an interrupt signal is generated or the message is treated as if it were an `M_DATA` message containing a single ASCII NUL character, depending on the state of the `BRKINT` flag.

`M_DATA`
: This message is normally processed using the standard `termio` input processing. If the `ICANON` flag is set, a single input record ("line") is accumulated in an internal buffer and sent upstream when a line-terminating character is received. If the `ICANON` flag is not set, other input processing is performed and the processed data is passed upstream.

  If output is to be stopped or started as a result of the arrival of characters (usually <CNTRL-Q> and <CNTRL-S>), `M_STOP` and `M_START` messages are sent downstream. If the `IXOFF` flag is set and input is to be stopped or started as a result of flow control considerations, `M_STOPI` and `M_STARTI` messages are sent downstream.

  `M_DATA` messages are sent downstream, as necessary, to perform echoing.

  If a signal is to be generated, an `M_FLUSH` message with a flag byte of `FLUSHR` is placed on the read queue. If the signal is also to flush output, an `M_FLUSH` message with a flag byte of `FLUSHW` is sent downstream.

`M_CTL`
: If the size of the data buffer associated with the message is the size of `struct iocblk, ldterm` will perform functional negotiation to determine where the `termio(7)` processing is to be done. If the command field of the `iocblk` structure (`ioc_cmd`) is set to `MC_NO_CANON`, the input canonical processing normally performed on `M_DATA` messages is disabled and those messages are passed upstream unmodified; this is for the use of modules or drivers that perform their own input processing, such as a pseudo-terminal in `TIOCREMOTE` mode connected to a program that performs this processing. If the command is `MC_DO_CANON`, all input processing is enabled. If the command is `MC_PART_CANON`, then an `M_DATA` message containing a `termios` structure is expected to be attached to the original `M_CTL` message. The `ldterm` module will examine the `iflag, oflag`, and `lflag` fields of the `termios` structure and from then on will process only those flags which have not been turned ON. If none of the above commands are found, the message is ignored; in any case, the message is passed upstream.

`M_FLUSH`

The read queue of the module is flushed of all its data messages and all data in the record being accumulated is also flushed. The message is passed upstream.

M_IOCACK
: The data contained within the message, which is to be returned to the process, is augmented if necessary, and the message is passed upstream.

All other messages are passed upstream unchanged.

Write side behavior

Various types of STREAMS messages are processed as follows:

M_FLUSH
: The write queue of the module is flushed of all its data messages and the message is passed downstream.

M_IOCTL
: The function of this ioctl is performed and the message is passed downstream in most cases. The TCFLSH and TCXONC ioctls can be performed entirely in the ldterm module, so the reply is sent upstream and the message is not passed downstream.

M_DATA
: If the OPOST flag is set, or both the XCASE and ICANON flags are set, output processing is performed and the processed message is passed downstream along with any M_DELAY messages generated. Otherwise, the message is passed downstream without change.

All other messages are passed downstream unchanged.

IOCTLS

The following ioctls are processed by the ldterm module. All others are passed downstream. EUC_WSET and EUC_WGET are I_STR ioctl calls whereas other ioctls listed here are TRANPARENT ioctls.

TCGETS/TCGETA
: The message is passed downstream; if an acknowledgment is seen, the data provided by the driver and modules downstream is augmented and the acknowledgement is passed upstream.

TCSETS/TCSETSW/TCSETSF/TCSETA/TCSETAW/TCSETAF
: The parameters that control the behavior of the ldterm module are changed. If a mode change requires options at the stream head to be changed, an M_SETOPTS message is sent upstream. If the ICANON flag is turned on or off, the read mode at the stream head is changed to message-nondiscard or byte-stream mode, respectively. If the TOSTOP flag is turned on or off, the tostop mode at the stream head is turned on or off, respectively.

TCFLSH
: If the argument is 0, an M_FLUSH message with a flag byte of FLUSHR is sent downstream and placed on the read queue. If the argument is 1, the write queue is flushed of all its data messages and an M_FLUSH message with a flag byte of FLUSHW is sent upstream and downstream. If the argument is 2, the write queue is flushed of all its data messages and an M_FLUSH message with a flag byte of FLUSHRW is sent downstream and placed on the read queue.

TCXONC
: If the argument is 0 and output is not already stopped, an M_STOP message is sent downstream. If the argument is 1 and output is stopped, an M_START message is sent downstream. If the argument is 2 and input is not already stopped, an M_STOPI message is sent downstream. If the argument is 3 and input is stopped, an M_STARTI message is sent downstream.

TCSBRK
: The message is passed downstream, so the driver has a chance to drain the data and then send an M_IOCACK message upstream.

EUC_WSET
: This call takes a pointer to an eucioc structure, and uses it to set the EUC line discipline's local definition for the code set widths to be used for subsequent operations. Within the stream, the line

discipline may optionally notify other modules of this setting via `M_CTL` messages.

`EUC_WGET`
: This call takes a pointer to an `eucioc` structure, and returns in it the EUC code set widths currently in use by the EUC line discipline.

`LSDMAP/LDGMAP/LDNMAP`
: These calls are used to load, read or delete a `mapchan` table. `LSDMAP` and `LDGMAP` require a pointer to an `emtab` structure of the size `E_TABSZ`, which contains the table to be loaded or to which the table being loaded is written. These calls are used by the `mapchan(1M)` command.

SEE ALSO

`mapchan(1M)`, `termios(3C)`, `mapchan(4)`, `console(7)`, `ports(7)`, `termio(7)`.

Programmer's Guide: STREAMS

System Administrator's Reference Manual

log(7)

NAME

log - interface to STREAMS error logging and event tracing

DESCRIPTION

log is a STREAMS software device driver that provides an interface for console logging and for the STREAMS error logging and event tracing processes [strerr(1M) , strace(1M)]. log presents two separate interfaces: a function call interface in the kernel through which STREAMS drivers and modules submit log messages; and a subset of ioctl(2) system calls and STREAMS messages for interaction with a user level console logger, an error logger, a trace logger, or processes that need to submit their own log messages.

Kernel Interface

log messages are generated within the kernel by calls to the function strlog:

```
strlog(mid, sid, level, flags, fmt, arg1, ...)
short mid, sid;
char level;
ushort flags;
char *fmt;
unsigned arg1;
```

Required definitions are contained in <sys/strlog.h>, <sys/log.h>, and <sys/syslog.h>. *mid* is the STREAMS module id number for the module or driver submitting the log message. *sid* is an internal sub-id number usually used to identify a particular minor device of a driver. *level* is a tracing level that allows for selective screening out of low priority messages from the tracer. *flags* are any combination of SL_ERROR (the message is for the error logger), SL_TRACE (the message is for the tracer), SL_CONSOLE (the message is for the console logger), SL_FATAL (advisory notification of a fatal error), and SL_NOTIFY (request that a copy of the message be mailed to the system administrator). *fmt* is a printf(3S) style format string, except that %s, %e, %E, %g, and %G conversion specifications are not handled. Up to NLOGARGS (currently 3) numeric or character arguments can be provided.

User Interface

log is opened via the clone interface, /dev/log. Each open of /dev/log obtains a separate stream to log. In order to receive log messages, a process must first notify log whether it is an error logger, trace logger, or console logger via a STREAMS I_STR ioctl call (see below). For the console logger, the I_STR ioctl has an ic_cmd field of I_CONSLOG, with no accompanying data. For the error logger, the I_STR ioctl has an ic_cmd field of I_ERRLOG, with no accompanying data. For the trace logger, the ioctl has an ic_cmd field of I_TRCLOG, and must be accompanied by a data buffer containing an array of one or more struct trace_ids elements. Each trace_ids structure specifies an *mid*, *sid*, and *level* from which message will be accepted. strlog will accept messages whose *mid* and *sid* exactly match those in the trace_ids structure, and whose level is less than or equal to the level given in the trace_ids structure. A value of −1 in any of the fields of the trace_ids structure indicates that any value is accepted for that field.

Once the logger process has identified itself via the ioctl call, log will begin sending up messages subject to the restrictions noted above. These messages are obtained via the getmsg(2) system call. The control part of this message contains a log_ctl structure, which specifies the *mid*, *sid*, *level*, *flags*, time in ticks since boot that the message was submitted, the corresponding time in seconds since Jan. 1, 1970, a sequence number, and a priority. The time in seconds since 1970 is provided so that the date and time of the message can be easily computed, and the time in ticks since boot is provided so that the relative timing of log messages can be determined.

The priority is comprised of a priority code and a facility code, found in <sys/syslog.h>. If SL_CONSOLE is set in *flags*, the priority code is set as follows. If SL_WARN is set, the priority code is set to LOG_WARNING. If

SL_FATAL is set, the priority code is set to LOG_CRIT. If SL_ERROR is set, the priority code is set to LOG_ERR. If SL_NOTE is set, the priority code is set to LOG_NOTICE. If SL_TRACE is set, the priority code is set to LOG_DEBUG. If only SL_CONSOLE is set, the priority code is set to LOG_INFO. Messages originating from the kernel have the facility code set to LOG_KERN. Most messages originating from user processes will have the facility code set to LOG_USER.

Different sequence numbers are maintained for the error and trace logging streams, and are provided so that gaps in the sequence of messages can be determined (during times of high message traffic some messages may not be delivered by the logger to avoid hogging system resources). The data part of the message contains the unexpanded text of the format string (null terminated), followed by NLOGARGS words for the arguments to the format string, aligned on the first word boundary following the format string.

A process may also send a message of the same structure to log, even if it is not an error or trace logger. The only fields of the log_ctl structure in the control part of the message that are accepted are the *level*, *flags*, and *pri* fields; all other fields are filled in by log before being forwarded to the appropriate logger. The data portion must contain a null terminated format string, and any arguments (up to NLOGARGS) must be packed one word each, on the next word boundary following the end of the format string.

ENXIO is returned for I_TRCLOG ioctls without any trace_ids structures, or for any unrecognized I_STR ioctl calls. Incorrectly formatted log messages sent to the driver by a user process are silently ignored (no error results).

Processes that wish to write a message to the console logger may direct their output to /dev/conslog, using either write(2) or putmsg(2).

EXAMPLES

Example of I_ERRLOG notification.

```
struct strioctl ioc;
ioc.ic_cmd = I_ERRLOG;
ioc.ic_timout = 0; /* default timeout (15 secs.) */
ioc.ic_len = 0;
ioc.ic_dp = NULL;
ioctl(log, I_STR, &ioc);
```

Example of I_TRCLOG notification.

```
struct trace_ids tid[2];
tid[0].ti_mid = 2;
tid[0].ti_sid = 0;
tid[0].ti_level = 1;
tid[1].ti_mid = 1002;
tid[1].ti_sid = -1;     /* any sub-id will be allowed */
tid[1].ti_level = -1;   /* any level will be allowed */
ioc.ic_cmd = I_TRCLOG;
ioc.ic_timout = 0;
ioc.ic_len = 2 * sizeof(struct trace_ids);
ioc.ic_dp = (char *)tid;
ioctl(log, I_STR, &ioc);
```

Example of submitting a log message (no arguments).

```
struct strbuf ctl, dat;
struct log_ctl lc;
char *message = "Don't forget to pick up some milk
                 on the way home";
ctl.len = ctl.maxlen = sizeof(lc);
ctl.buf = (char *)&lc;
dat.len = dat.maxlen = strlen(message);
dat.buf = message;
lc.level = 0;
lc.flags = SL_ERROR|SL_NOTIFY;
putmsg(log, &ctl, &dat, 0);
```

FILES

/dev/log
/dev/conslog
<sys/log.h>
<sys/strlog.h>
<sys/syslog.h>

SEE ALSO

strace(1M), strerr(1M), getmsg(2), intro_prm2(2), putmsg(2), write(2), clone(7).
Programmer's Guide: STREAMS

System Administrator's Reference Manual

mapkey(7)

NAME

`mapkey` - mechanism for PC terminals

DESCRIPTION

Keyboard mapping

The keyboard mapping structure is defined in `/usr/include/sys/mapkey.h`. Each key can be in any of ten states. The first eight states are:

- `BASE`
- `CTRL-SHIFT`
- `SHIFT`
- `ALT-SHIFT`
- `CTRL`
- `ALT-CTRL`
- `ALT`
- `ALT-CTRL-SHIFT`

The other two states relate to two special bytes. The first byte defines a special state. The bits in it indicate whether the key has been identified as a special key in one or more of the first eight states. The second byte consists of one of four possible codes, `C`, `N`, `B` or `O`, indicating how the lock keys affect the key in question.

The table below describes the default keyboard mapping values. All values other than special keywords (described further below) are ASCII character values.

Column heading
: **Description**

`SCAN CODE`
: This column contains the scan code generated by the keyboard hardware when a key is pressed. The table does not show the scan code generated when a key is released.

`BASE` This column shows the base value of a keystroke.

`SHIFT`
: This column shows the value of a keystroke when the key is combined with `SHIFT`.

`LOCK` This column indicates the lock keys which affect the key in question:

- `C` stands for `CAPSLOCK`
- `N` stands for `NUMLOCK`
- `B` stands for both
- `O` stands for lock function off

The other columns contain the values of keys pressed together with a combination of the `CTRL`, `ALT` and `SHIFT` keys.

SCAN CODE	BASE	SHIFT	CTRL	CTRL SHIFT
0	nop	nop	nop	nop
1	esc	esc	nop	nop
2	'1'	'!'	nop	nop

3	'2'	'"'	nop	nop
4	'3'	0247	nop	nop
5	'4'	'$'	nop	nop
6	'5'	'%'	nop	nop
7	'6'	'&'	nop	nop
8	'7'	'/'	nop	nop
9	'8'	'('	nop	nop
10	'9'	')'	nop	nop
11	'0'	'='	nop	nop
12	0337	'?'	nop	nop
13	0264	'`'	nop	nop
14	bs	bs	del	nop
15	ht	fkey12	nop	nop
16	'q'	'Q'	dc1	dc1
17	'w'	'W'	etb	etb
18	'e'	'E'	enq	enq
19	'r'	'R'	dc2	dc2
20	't'	'T'	dc4	dc4
21	'z'	'Z'	em	em
22	'u'	'U'	nak	nak
23	'i'	'I'	ht	ht

SCAN CODE	BASE	SHIFT	CTRL	CTRL SHIFT
24	'o'	'O'	si	si
25	'p'	'P'	dle	dle
26	0374	0334	nop	nop
27	'+'	'*'	nop	nop
28	cr	cr	nop	nop
29	lctrl	lctrl	lctrl	lctrl
30	'a'	'A'	soh	soh
31	's'	'S'	dc3	dc3
32	'd'	'D'	eot	eot
33	'f'	'F'	ack	ack
34	'g'	'G'	bel	bel
35	'h'	'H'	bs	bs
36	'j'	'J'	nl	nl

37	'k'	'K'	vt	vt
38	'l'	'L'	np	np
39	0366	0326	nop	nop
40	0344	0304	nop	nop
41	'^'	0260	0036	0036
42	lshift	lshift	lshift	lshift
43	'#'	'\'	nop	nop
44	'y'	'Y'	sub	sub
45	'x'	'X'	can	can
46	'c'	'C'	etx	etx
47	'v'	'V'	syn	syn
48	'b'	'B'	stx	stx
49	'n'	'N'	so	so
50	'm'	'M'	cr	cr
51	','	';'	nop	nop
52	'.'	':'	nop	nop
53	'-'	'_'	nop	0037
54	rshift	rshift	rshift	rshift
55	'*'	'*'	nop	nop
56	lalt	lalt	lalt	lalt
57	' '	' '	nop	nop
58	clock	clock	clock	clock
59	fkey0	fkey12	fkey24	fkey36
60	fkey1	fkey13	fkey25	fkey37
61	fkey2	fkey14	fkey26	fkey38
62	fkey3	fkey15	fkey27	fkey39
63	fkey4	fkey16	fkey28	fkey40

SCAN CODE	BASE	SHIFT	CTRL	CTRL SHIFT
64	fkey5	fkey17	fkey29	fkey41
65	fkey6	fkey18	fkey30	fkey42
66	fkey7	fkey18	fkey31	fkey43
67	fkey8	fkey20	fkey32	fkey44
68	fkey9	fkey21	fkey33	fkey45
69	nlock	nlock	nlock	nlock
70	slock	slock	del	nop

71	fkey48	'7'	'7'	'7'
72	fkey49	'8'	'8'	'8'
73	fkey50	'9'	'9'	'9'
74	fkey51	'-'	nop	nop
75	fkey52	'4'	'4'	'4'
76	fkey53	'5'	'5'	'5'
77	fkey54	'6'	'6'	'6'
78	fkey55	'+'	nop	nop
79	fkey56	'1'	'1'	'1'
80	fkey57	'2'	'2'	'2'
81	fkey58	'3'	'3'	'3'
82	fkey59	'0'	'0'	'0'
83	del	','	','	','
84	nop	nop	nop	nop
85	nop	nop	nop	nop
86	'<'	'>'	nop	nop
87	fkey10	fkey22	fkey34	fkey46
88	fkey11	fkey23	fkey35	fkey47
89	nop	nop	nop	nop
90	nop	nop	nop	nop
91	nop	nop	nop	nop
92	nop	nop	nop	nop
93	nop	nop	nop	nop
94	nop	nop	nop	nop
95	nop	nop	nop	nop
96	nop	nop	nop	nop
97	nop	nop	nop	nop
98	nop	nop	nop	nop
99	nop	nop	nop	nop
100	nop	nop	nop	nop
101	nop	nop	nop	nop
102	nop	nop	nop	nop
103	nop	nop	nop	nop

SCAN CODE	BASE	SHIFT	CTRL	CTRL SHIFT
104	nop	nop	nop	nop

105	nop	nop	nop	nop
106	nop	nop	nop	nop
107	nop	nop	nop	nop
108	nop	nop	nop	nop
109	nop	nop	nop	nop
110	nop	nop	nop	nop
111	nop	nop	nop	nop
112	nop	nop	nop	nop
113	nop	nop	nop	nop
114	nop	nop	nop	nop
115	nop	nop	nop	nop
116	nop	nop	nop	nop
117	nop	nop	nop	nop
118	nop	nop	nop	nop
119	nop	nop	nop	nop
120	nop	nop	nop	nop
121	nop	nop	nop	nop
122	nop	nop	nop	nop
123	nop	nop	nop	nop
124	nop	nop	nop	nop
125	nop	nop	nop	nop
126	nop	nop	nop	nop
127	nop	nop	nop	nop
128	rctrl	rctrl	rctrl	rctrl
129	ralt	ralt	ralt	ralt
130	fkey59	fkey59	nop	nop
131	del	del	0037	0037
132	fkey48	fkey48	nop	nop
133	fkey56	fkey56	nop	nop
134	fkey50	fkey50	nop	nop
135	fkey58	fkey58	nop	nop
136	fkey54	fkey54	nop	nop
137	fkey52	fkey52	nop	nop
138	fkey49	fkey49	nop	nop
139	fkey57	fkey57	nop	nop
140	'/'	'/'	nop	nop
141	cr	cr	nop	nop

The next table lists the values of the special keywords used in the previous table. These keywords were used above purely on grounds of readability; in the actual keyboard mapping table a special keyword is formed on the basis of its value and the corresponding bit in the ninth state byte for a special state is set simultaneously.

Name:Value:Function
nop:0:No operation – the keystroke has no effect
lshift:2:left-hand SHIFT key
rshift:3:right-hand SHIFT key
clock:4:Caps lock (uppercase letters only)
nlock:5:Numeric lock (numeric values only)
slock:6:Scroll lock (scrolling on/off)
alt:7:ALT key
btab:8:Reset key – generates a fixed sequence (ESC[Z)
ctrl:9:Control key
lalt:10:Left-hand ALT key
ralt:11:Right-hand ALT key
lctrl:12:Left-hand control key
rctrl:13:Right-hand control key
fkey1:27:Function key #1
.::.
.::.
.::.
fkey96:122:Function key #96
sysreq:123:System request
brk:124:Break key

escn:125:Generate escape sequence ESC N *x*, where *x* is the value of the scan code without the ALT key esco:126:Generate escape sequence ESC O *x*, where *x* is the value of the scan code without the ALT key escl:127:Generate escape sequence ESC L *x*, where *x* is the value of the scan code without the ALT key □

The next table lists the names and decimal values of the ASCII characters in the previous table. Numeric constants are replaced by names to make the scan code table easier to read. Only the decimal values are placed in the `ioctl` buffer. These values are taken from `ascii(5)`.

Name:Value:Name:Value:Name:Value:Name:Value:Name:Value		
nul:0:ht:9:dc2:18:esc:27 soh:1:nl:10:dc3:19:fs:28 stx:2:vt:11:dc4:20:gs:29 etx:3:np:12:nak:21:rs:30 eot:4:cr:13:syn:22:ns:31 enq:5:so:14:etb:23:del:127 ack:6:si:15:can:24 bel:7:dle:16:em:25 bs:8:dc1:17:sub:26		

Function key assignments

The string mapping table is a 512-byte area (`typedef strmap_t`). This area contains null-terminated strings which redefine function keys. The first null-terminated string is assigned to the first function key, the second string to the second function key, and so on.

There is no restriction on the length of a given string as long as the whole table, including the nulls, is no more

than 512 bytes long. To set a string to null, extra null characters are added. The following table illustrates the default function key assignments.

String key values

"\033[M":Functio n #1
"\033[N":Functio n #2
"\033[O":Functio n #3
"\033[P":Functio n #4
"\033[Q":Functio n #5
"\033[R":Functio n #6
"\033[S":Functio n #7
"\033[T":Functio n #8
"\033[U":Functio n #9
"\033[V":Functio n #10
"\033[W":Functio n #11
"\033[X":Functio n #12
"\033[Y":Shift Function #1
"\033[Z":Shift Function #2
"\033[a":Shift Function #3
"\033[b":Shift Function #4
"\033[c":Shift Function #5
"\033[d":Shift Function #6
"\033[e":Shift Function #7
"\033[f":Shift Function #8
"\033[g":Shift Function #9
"\033[h":Shift Function #10

```
"\033[i":Shift
Function #11
"\033[j":Shift
Function #12
"\033[k":Control
Function #1
"\033[l":Control
Function #2
"\033[m":Control
Function #3
"\033[n":Control
Function #4
"\033[o":Control
Function #5
"\033[p":Control
Function #6
```

```
"\033[q":Control
Function #7
"\033[r":Control
Function #8
"\033[s":Control
Function #9
"\033[t":Control
Function #10
"\033[u":Control
Function #11
"\033[v":Control
Function #12
"\033[w":Ctrl/Sh
ft Function #1
"\033[x":Ctrl/Sh
ft Function #2
"\033[y":Ctrl/Sh
ft Function #3
"\033[z":Ctrl/Sh
ft Function #4
"\033[@":Ctrl/Sh
ft Function #5
"\033[[":Ctrl/Sh
ft Function #6
"\033[\":Ctrl/Sh
ft Function #7
"\033[]":Ctrl/Sh
ft Function #8
"\033[^":Ctrl/Sh
ft Function #9
"\033[_":Ctrl/Sh
ft Function #10
```

```
"\033[`":Ctrl/Sh
ft Function #11
"\033[{":Ctrl/Sh
ft Function #12
"\033[H":Home
"\033[A":Up
arrow
"\033[I":Page up
"-":-
"\033[D":Left
arrow
"\033[E":5
"\033[C":Right
arrow
"+":+
"\033[F":End
"\033[B":Down
arrow
"\033[G":Page
down
"\033[L":Insert
```

\{\

```
"\033[M":Fun
ction #1
"\033[N":Fun
ction #2
"\033[O":Fun
ction #3
"\033[P":Fun
ction #4
"\033[Q":Fun
ction #5
"\033[R":Fun
ction #6
"\033[S":Fun
ction #7
"\033[T":Fun
ction #8
"\033[U":Fun
ction #9
"\033[V":Fun
ction #10
"\033[W":Fun
ction #11
"\033[X":Fun
ction #12
```

```
"\033[Y":Shi
ft Function
#1
"\033[Z":Shi
ft Function
#2
"\033[a":Shi
ft Function
#3
"\033[b":Shi
ft Function
#4
"\033[c":Shi
ft Function
#5
"\033[d":Shi
ft Function
#6
"\033[e":Shi
ft Function
#7
```

```
"\033[f":Shi
ft Function
#8
"\033[g":Shi
ft Function
#9
"\033[h":Shi
ft Function
#10
"\033[i":Shi
ft Function
#11
"\033[j":Shi
ft Function
#12
"\033[k":Con
trol
Function #1
"\033[l":Con
trol
Function #2
"\033[m":Con
trol
Function #3
"\033[n":Con
trol
Function #4
"\033[o":Con
trol
```

```
Function #5
"\033[p":Con
trol
Function #6
"\033[q":Con
trol
Function #7
"\033[r":Con
trol
Function #8
"\033[s":Con
trol
Function #9
"\033[t":Con
trol
Function #10
"\033[u":Con
trol
Function #11
"\033[v":Con
trol
Function #12
"\033[w":Ctr
l/Shft
Function #1
"\033[x":Ctr
l/Shft
Function #2
"\033[y":Ctr
l/Shft
Function #3
"\033[z":Ctr
l/Shft
Function #4
"\033[@":Ctr
l/Shft
Function #5
"\033[[":Ctr
l/Shft
Function #6
"\033[\":Ctr
l/Shft
Function #7
"\033[]":Ctr
l/Shft
Function #8
"\033[^":Ctr
l/Shft
Function #9
"\033[_":Ctr
l/Shft
```

```
Function #10
"\033[`":Ctr
l/Shft
Function #11
"\033[{":Ctr
l/Shft
Function #12
"\033[H":Hom
e
"\033[A":Up
arrow
"\033[I":Pag
e up
"-":-
"\033[D":Lef
t arrow
"\033[E":5
"\033[C":Rig
ht arrow
"+":+
"\033[F":End
"\033[B":Dow
n arrow
"\033[G":Pag
e down
"\033[L":Ins
ert
```

ioctl calls

`KDGKBMODE`

This call checks the current keyboard mode. It returns one of the following values as defined in `/usr/include/sys/mapkey.h`:

```
#define K_RAW  0x00 /* send scan code for line */
#define K_XLATE 0x01 /* translate to ASCII */
```

A pointer to a variable in which the value is to be stored must be passed.

`KDSKBMODE`

This call sets the current keyboard mode. The call argument is either `K_RAW` or `K_XLATE`. In raw mode the program can read the raw up/down scan codes from the keyboard. In translation mode the translation tables are used to generate the appropriate scan code.

`KDGKBENT`

This call reads one of the entries from the translation table. The call argument is the address of one of the following structures as defined in `/usr/include/mapkey/kd.h`. The first two fields are filled.

```
struct kbentry {
       unchar kb_table;     /* table to read */
       unchar kb_index;     /* entry in table */
       ushort kb_value;     /* value to read/set */
 };
```

The following are the allowable values for `kb_table`:

```
#define K_NORMTAB  0x00     /* BASE */
#define K_SHIFTTAB  0x01    /* SHIFT */
#define K_ALTTAB  0x02      /* ALT */
#define K_ALTSHIFTTAB 0X03  /* SHIFT ALT */
```

This `ioctl` fetches the specified entry from the translation table and returns it in the third field.

`KDSKBENT`
: This call sets an entry in one of the translation tables. It uses the same structure as the `KDGKBENT ioctl`, with the third field being supplied with the value which is to be entered in the translation table. This is a way of fully or partly readdressing the keyboard.

`GIO_KEYMAP`
: This call fetches the entire keyboard mapping table from the kernel. The structure of the argument is defined in `/usr/include/sys/mapkey.h`.

`PIO_KEYMAP`
: This call sets the entire keyboard mapping table. The structure of the argument is defined in `/usr/include/sys/mapkey.h`.

`GIO_STRMAP`
: This call fetches the string mapping table from the kernel. The structure of the argument is defined in `/usr/include/sys/mapkey.h`.

`PIO_STRMAP`
: This call sets the string mapping table. The structure of the argument is defined in `/usr/include/sys/mapkey.h`.

`GETFKEY`
: Fetches the current definition of a function key. The argument for this call is the address of one of the following structures as defined in `/usr/include/sys/mapkey.h`:

```
struct fkeyarg {
       unsigned int keynum;
       char         keydef [MAXFK]; /* via comcrt.h from ioctl.h */
       char         flen;
};
```

The function key number must be passed in `keynum` (see `arg` structure above). After execution of the `ioctl`, the key sequence currently assigned to the key is returned in `keydef`, and the length of the sequence is returned in `flen`.

`SETFKEY`
: Assigns a function key to a given key sequence. The call uses the same structure as the `GETFKEY ioctl`. The number of the function key must be passed in `keynum`, the key sequence in `keydef`, and the length of the key sequence (number of characters) in `flen`.

FILES

`/usr/lib/keyboard/*`

SEE ALSO

`mapkey(1M)` .

System Administrator's Reference Manual

mem(7)

NAME

`mem, kmem` - core memory

DESCRIPTION

The `/dev/mem` file is a special file that is an image of the core memory of the computer. It may be used, for example, to examine, and even to patch the system.

Byte addresses in `/dev/mem` are interpreted as memory addresses. References to non-existent locations cause errors to be returned.

Examining and patching device registers is likely to lead to unexpected results when read-only or write-only bits are present.

The file `/dev/kmem` is the same as `/dev/mem` except that kernel virtual memory rather than physical memory is accessed.

NOTES

Some of `/dev/kmem` cannot be read because of write-only addresses or unavailable memory addresses.

FILES

`/dev/mem`
`/dev/kmem`

System Administrator's Reference Manual

mtio(7)

NAME

`mtio` - magnetic tape interface

DESCRIPTION

Magnetic tape filenames have the following format:

`/dev/ios`*b*`/`[`r`]`stape`*ppu*[`c`][`h`][`n`][`o`][`v`][`g`]

Refer to the `stape(7)` manual page for more information.

The actions taken by the tape drive at close depend on how the drive was opened and what it was used for.

The default tape drive (for example, `/dev/ios0/rstape004`) will write 2 file marks (1 file mark if it is a QIC tape device) and rewind to the Beginning-of-Tape (BOT) at device close if the device was opened for writing. If reading the tape, it will be rewound to BOT at close. The tape will be written using the drive's default density.

The high density tape (for example, `/dev/ios0/rstape004h`) will act the same as the default tape detailed above, except the tape will be written using the drive's high density.

The no-rewind tape (for example, `/dev/ios0/rstape004hn` or `/dev/ios0/rstape004n`) behaves very differently than the default tape. If writing the tape, 2 file marks will be written to the tape at close (1 file mark if it's a QIC drive) and the tape will be positioned *after* the first file mark at close. If reading the tape, it will be positioned *after* the file mark. So, if you are reading, do a close and the tape is not currently just *after* a file mark, the driver will do a forward space file to position the tape after the next file mark. If you are reading, do a close, and the tape is currently just after a file mark, the position will not be changed. The density the tape is written will depend on whether the `h` device was used.

The OSX no-rewind tape (for example, `/dev/ios0/rstape004o`) will behave like the standard no-rewind tape with the difference, that the tape will *not* be positioned after reads.

Currently, only the raw (character) interface is supported by the IOS tape subsystem. An attempt to open the IOS block tape device will result in `EINVAL` error returned by the open routine.

Reading a tape as a block device requires that the tape was written using 2048-byte records. This can be done with the character device. Then, if supported, the block device can be read. The tape will be read using 2048-byte record size. Using this method, the system makes it possible, if inefficient, to treat the block tape device like any other file. Seeks have their usual meaning and it is possible to read a byte at a time. Writing in very small units is inadvisable, however, because it tends to create monstrous record gaps in variable block-sized tapes and can have a negative effect on performance.

Each `read` or `write` call reads or writes the next record on the tape. In the `write` case, the record has the same length as the buffer given. During a `read`, the record size is passed back as the number of bytes read, provided it is no greater than the buffer size; if the record is longer, an error is indicated. In raw tape, I/O seeks are ignored. A zero byte count is returned when a tape mark is read. Subsequent reads will continue until another file mark is encountered or another error is detected (such as End-of-Tape).

A number of `ioctl` operations are available on raw magnetic tape. The following definitions are from `</usr/include/sys/mtio.h>`:

```
/*
 * Structures and definitions for mag tape io control commands
 */
/* structure for MTIOCTOP - mag tape op command */
struct mtop
{
    short   mt_op;     /* operations defined below */
    daddr_t mt_count;  /* how many of them */
};
/* operations */
```

```
#define MTWEOF    0   /* write an end-of-file record */
#define MTFSF     1   /* forward space file */
#define MTBSF     2   /* backward space file */
#define MTFSR     3   /* forward space record */
#define MTBSR     4   /* backward space record */
#define MTREW     5   /* rewind */
#define MTOFFL    6   /* rewind and put the drive offline */
#define MTNOP     7   /* no operation, sets status only */
#define MTRET     8   /* retension operation */
#define MTRST     9   /* reset operation */
#define MTBLKSIZE 10  /* get block size */
#define MTONL     11  /* put drive online */
#define MTAPP     12  /* space to end of data */
#define MTERA     13  /* erase fixed length of tape */
#define MTRETEN   14  /* retension (Cartridge tape only) */
#define MTWSM     15  /* currently unused */
#define MTFSSM    16  /* currently unused */
#define MTBSSM    17  /* currently unused */
#define MTRELOF   18  /* permanently reserved */
#define MTSWON    19  /* enable a specific feature   */
#define MTSWOFF   20  /* disable a specific feature */
#define MTCAPA    21  /* establish the capacity */
/* structure for MTIOCGET - mag tape get status command */
struct mtget
{
    short   mt_type;    /* type of magtape device */
    /* the following two registers are grossly device dependent */
    short   mt_dsreg;   /* "drive status" register */
    short   mt_erreg;   /* "error" register */
    /* end device-dependent registers */
    daddr_t mt_resid;   /* residual count */
    /* the following two are not yet implemented */
    daddr_t mt_fileno;  /* file number of current position */
    daddr_t mt_blkno;   /* block number of current position */
    /* end not yet implemented */
};
/*
* Constants for mt_type byte
*/
#define MT_TIOC 0x08  /* IOC/Tapemaster tape drive */
#define MT_TTPE 0x09  /* TPE/Pertec tape drive */
#define MT_TIOS 0x0a  /* IOS/Tape drive */
/* mag tape io control commands */
#define MTIOC           (ERRORtERROR<<8)
#define MTIOCTOP        (MTIOC|1)  /* do a mag tape op */
#define MTIOCGET        (MTIOC|2)  /* get tape status */
#define MTIOCPOSTEOTOK  (MTIOC|3)  /* allow post-eot writes */
#define MTIOCEOTCOMPAT  (MTIOC|4)  /* enable EOT compatibility */
#define MTIOCGETIOSTAT  (MTIOC|5)  /* get iotime */
#define MTIOSGETERROR   (MTIOC|6)  /* get error stats */
```

If the node of a tape device is opened, the tape driver reserves this device for the duration of the job and frees it again when the job is finished. High-performance tape drives sometimes have two independent SCSI buses which enable connection to two different systems at the same time. The `ioctl` command `MTRELOF` was introduced for this purpose so that one of the connected systems can permanently reserve the tape drive for the duration of several jobs. The `ioctl` command `MTRELOF` has its own structure for communicating with the operating system. Apart from permanently reserving and freeing the drive, the reservation status of a tape device can also be queried.

```
/* Structure of MTRELOF command for permanently reserving tape devices */
```

```
struct  resrel  {
      short   rescmd;         /* command for reserving, querying */
                                /* or determining the status */
      short   stat;           /* return of reservation status */
};
/* Definitions of commands and returned status */
#define RES_RELEASE         0   /* command for freeing a reservation */
#define RES_RESERVE         1   /* command for permanent reservation */
#define RES_STATUS          2   /* command for querying the reservation status */
#define RES_RESERVE_LOCAL   3   /* status in resrel.stat, if the device was */
                                /* reserved from the local system */
#define RES_RESERVE_REMOTE  4   /* status in resrel.stat, if the device was */
                                /* reserved from the remote system */
#define RES_UNDEF           5   /* status in resrel.stat, if the reservation */
                                /* status could not be determined */
```

If a drive could not be reserved or freed, the `errno` variable is set to the value `EIO` or `EACCES`. If the value is `EACCES` (no access permission), the device was reserved from another system.

The `ioctl MTCAPA` command can be used if an application is to establish the total capacity of a data medium and/or how much more data can be stored on this data medium.

If the `cmd` variable is set to the value `ENT_CAPA`, the total capacity of the medium is calculated and the result made available in the `cartcapa` variable. If `cmd` is set to the value `REM_CAPA`, the residual capacity of the medium is calculated and the result made available in the `remaincapa` variable.

```
/*
** Job field for the ioctl MTCAPA command for establishing
** the capacity of the cassette.
** Cartridge in KB.
*/
struct tapecapa {
   int     cmd;            /* Total or residual capacity */
   int     capa_stat;      /* Status following capacity check */
   long    cartcapa;       /* Total capacity in KB */
   long    remaincapa;     /* Residual capacity in KB */
   int     traversed;      /* Number of written segments */
   int     divider;        /* Total number of tape segments */
};
```

When the `ioctl MTCAPA` command has been executed, the `capa_stat` variable contains the following status information.

Value of `capa_stat`:
: Meaning:

`CAPA_GOT`
: Successful implementation

`CAPA_NOT_SUPPORT`
: The drive does not allow the respective capacity figure to be set.

`CAPA_CMD_UNKNOWN`
: The SCSI command, referred to in the profile structure of the drive in the `../sti/space.c` file, is not supported by the tape driver.

`CAPA_SENSE_FAILED`
: The SCSI command for ascertaining the capacity, which is referred to in the profile structure of the drive in the `../sti/space.c` file, could not be executed successfully.

```
/* Command definitions for the ioctl MTCAPA command*/
#define ENT_CAPA   1       /* Establish total capacity */
#define REM_CAPA   2       /* Establish residual capacity */
/* Constants for the ioctl MTCAPA command response */
#define CAPA_GOT            0  /* Successful */
```

```
#define CAPA_NOT_SUPPORT   -1  /* Capacity check not supported */
#define CAPA_CMD_UNKNOWN   -2  /* SCSI command for capacity check not supported *
/
#define CAPA_SENSE_FAILED  -3  /* SCSI command for capacity check unsuccessful */
```

The `traversed` and `divider` variables in the `tapecapa` structure have no meaning at present.

Due to technical limitations on individual tape drives, the precision of capacity measurement that can be achieved may vary. Measurements with the individual drives have shown that, depending on the drive type and the utilization level of the tape, inaccuracies in the range 0.5 to 8% may result.

In addition to the `ioctl`(2) operations represented here, all operations of the generic `sgen` driver are also available in the tape driver. This can be used for all tape devices with a connected media jukebox if the media jukebox and the tape device have a common SCSI-ID and can therefore be addressed via a single node. With the help of this interface, data media can, for example, be loaded in the actual drive and unloaded in the appropriate slot. If the tape is to be written to or read from directly by means of SCSI commands, the node with the ending "g" should be used (e.g. `/dev/ios0/rstape0`*XX*`g`). However, nodes with this ending only exist for high-performance drives. All required information on this `ioctl` interface can be found in the `sgenio`(7) manual page.

DIAGNOSTICS

The following is a list of all error conditions that may be encountered during tape operations. Each is followed by a short description, documenting the error returned to the user and the state of the tape after the error occurs.

`Beginning-of-Tape (BOT) encountered`
: During the last tape operation, the physical BOT marker (foil) was detected by the drive. The operation in progress is considered to have failed and an `EIO` error is returned. If the drive is accessed as the block device, detection of BOT sets internal state such that no further operations are allowed until the drive is closed and reopened. Since the tape location over the drive head is indeterminate after BOT, enforcement of this state prevents inaccurate seeking.

`Attempt to read blank tape or density mismatch`
: After beginning the current operation, the drive was unable to detect the presence of formatted data on the tape within a predetermined length. (*NOTE*: the actual length is entirely drive dependent.) `EIO` is returned to the user and all data found should be discarded. If the drive is accessed as the block device, detection of blank tape sets internal state such that no further operations are allowed until the drive is closed and reopened. Since the tape location over the drive head is indeterminate after this error, enforcement of this state prevents inaccurate seeking. This error may also be returned if the density selection is changed manually after a tape is mounted and the burst is read. Some tapes may appear blank to the drive after this happens, in which case a blank tape error is returned.

`Correctable error`
: During the last tape operation the drive encountered a correctable error condition. In the case of reads, the data is unaffected and no error is returned to the user, though an error is logged for the system. In the case of writes and wfms, an `EIO` is returned to the user only after a series of retry attempts are made. The tape is backspaced one record and a fixed length erase is performed and the I/O is tried again. If the retry sequence fails, the driver is left in a state that prevents further operations until the device is closed and reopened.

`Drive not ready (offline)`
: The drive has been taken physically offline during the last tape operation or before it began and an `ENXIO` is returned to the user. The condition is corrected by putting the drive back online before the next operation is initiated. If the drive is accessed as the block device, taking the drive offline marks the unit with a hard error and no further operations are allowed until the drive is closed and reopened. Since the tape location over the drive head is indeterminate after being offline, enforcement of this state prevents inaccurate seeking.

`Drive is already in use`
: The drive was already opened by another process. An `EBUSY` is returned to the user.

`Drive write protected`
: The drive was opened for writing but the tape had write protected (for example, no write ring or tab was set). An `ENXIO` is returned to the user and the drive remains unopened. The tape must be remounted with the write ring (or tab set correctly) or the drive may be opened for read-only access.

`End-of-File (EOF/FM) encountered`
: During the last tape operation a file mark was detected by the drive. This is translated as a zero byte return value at the user interface. The drive head is guaranteed to be at the beginning of the next record. If the drive is accessed as the block device, detection of a file mark sets the internal state such that no further forward reads or seeks are allowed until the tape is repositioned backwards (an `lseek` back) and additional data is written over the file mark.

`End-of-Tape (EOT) encountered`
: The behavior of tape I/O with regards to EOT is defined as follows:

  WRITES:

  Writes proceed until the EOT (reel termination) condition is detected. The software driver internally notes this condition, but assuming the write was complete (return byte counts equals request byte count), no error is returned to the user. Any subsequent writes cause the posting of an error (`ENOSPC`) to the user without moving the tape.

  READS:

  Reads proceed until the EOT condition is detected. No error is posted to the user. Reading may continue until input is satisfied, a filemark (zero byte return count) is encountered, or a blank tape error is returned by the driver.

  Previous versions of the operating system handled EOT in a different manner than that just described. As a result, some multi-reel backups, generated by programs that used EOT as a reel termination signal (e.g. `cpio`), may require special handling to be read back with the new subsystems. A new `ioctl`, `MTIOCEOTCOMPAT`, is implemented for this purpose. `MTIOCEOTCOMPAT` forces the software driver to generate an error in conjunction with the event. An argument is also required that directs the software driver to delay the EOT report by a corresponding number of reads. Typically, the value supplied should be a 1; however, tapes written on Kennedy tape drives at 6250 bpi may require a value of 2. This is due to the variation in which drives detect and report the EOT event and the effects of block size and density on that report. The `MTIOCEOTCOMPAT` `ioctl` may be issued any time after open and before EOT is encountered.

`FIFO over/underflow`
: During the last tape operation the drive FIFO was underrun (`write` was in progress) or overrun (`read` was in progress). An `EIO` is returned to the user only after a series of retry attempts are made. In both underrun and overrun cases, the tape is backspaced one record and the I/O is tried again. If the retry sequence fails, the driver is left in a state that prevents further operations until the device is closed and reopened.

`Hard error`
: During the last tape operation a media defect was encountered. An `EIO` is returned to the user only after a series of retry attempts are made. For reads, the tape is backspaced one record and the I/O is tried again. For writes, the tape is backspaced one record, a fixed length erase is performed and the I/O is tried again. If the retry sequence fails, the driver is left in a state that prevents further operations until the device is closed and reopened.

`Reservation conflict`
: A tape drive was accessed that had already been reserved by another system. Modern high-performance tape drives sometimes enable connection to two systems at the same time because they have two independent SCSI buses. If one of the systems has permanently reserved the tape drive, and if the second system tries to access the drive, a `Reservation conflict` is reported.

`Parity error`
: During the last tape operation, a parity error occurred during data transfer to the tape. An `EIO` is returned to the user only after a series of retry attempts are made. The tape is backspaced one record and the I/O is tried again. If the retry sequence fails, the driver is left in a state that prevents further

operations until the device is closed and reopened.

`Timeout on formatter – busy true`
: During a previous operation, the tape drive formatter had an error and left the interface in an undefined state. The next operation could not be initiated because one of the interface signals (formatter – busy) failed to return to the false state after a fixed timeout period. The most common cause of this error is an attempt to write blocks larger than can be accepted by the memory management firmware of a cached tape drive.

The status should be returned in a device independent format.

The SCSI status bytes are not returned for IOS tapes.

NOTES

Post-EOT I/O:
: Post-EOT I/O is a feature that allows the writing of non-UNIX format tapes that expect to place post-amble information (e.g. volume marks) on a tape after EOT is encountered. Tapes made with this type of format may now be read back. This feature works as follows:

  WRITES:

  After EOT is reached, further I/O is allowed under either of two conditions. In the first method, the tape unit in question must be opened as the no-rewind device (minor device number bit 2 is set). Closing the drive at EOT sets a state that allows post-EOT I/O on the subsequent open. Note that 1 filemark remains on the tape between the pre- and post-EOT sections of the tape. In the second method, no special device is required at open, but upon encountering the EOT condition, an `ioctl`, `MTIOCPOSTEOTOK`, must be issued. This allows further I/O without closing the device and leaves no interleaving filemark. In both cases, only eight more I/Os are allowed, after which an `EIO` is returned. This state remains set for all further writes and across opens until either of two actions occur: the drive is closed and/or rewinds, or a `MTBSR`/`MTBSF` `ioctl` is issued that repositions the drive before EOT.

  READS:

  Reads proceed until the EOT condition is detected. No error is posted to the user. Reading may continue until input is satisfied, a filemark (zero byte return count) is encountered, or a blank tape error is returned by the driver. *NOTE* that if the tape was created on a system like this one and the first method (closing and re-opening the no-rewind device at EOT; see the above paragraph) was used, the intervening filemark will be encountered. Programs designed to read these tapes should expect this event.

FILES

`/dev/ios`*b*`/[r]stape`*ppu*`[c][h][n][o][v][g]`

`/dev/rmt*`

SEE ALSO

`cpio`(1), `mt`(1), `tar`(1), `sgenio`(7), `stape`(7), `autoconf`(8), `tpsetup`(8).

System Administrator's Reference Manual

null(7)

NAME

`null` - the null file

DESCRIPTION

Data written to the null special file, `/dev/null`, is discarded.

Reads from a null special file always return 0 bytes.

FILES

`/dev/null`

System Administrator's Reference Manual

nvram(7)
RM400 only

NAME

`nvram` - environment variables of the PROM monitor and of theSASH

DESCRIPTION

Contents of the non-volatile RAM

This section describes all the environment variables in detail. The variables are listed in order of their occurrence in volatile RAM. The location of the variables stored in non-volatile RAM can be seen in the section **Locations of contents in non-volatile RAM**.

Base address of NVRAM: `0x1c080000`

`netaddr`

This variable consists of 4 bytes, containing the Internet address of the RM400. If you want to send data via TCP/IP (or other protocols using the Internet address), you must provide an Internet address for your RM400 (which should be unique within your network). Your name server can then translate your RM400 Ethernet address (stored in the Ethernet PROM and unique worldwide) to the RM400 Internet address and vice versa (make sure your name daemon is running).

Offset:
`0x00000000`

`lbaud` This byte contains the baud rate for communication with the tty at port `tty(0)`.

Offset:
`0x00000004`

Valid baud rates:
`300`, `600`, `1200`, `2400`, `4800`, `9600`, `19200` Bit/sec

Default value:
`9600`

`rbaud` This byte contains the baud rate for communication with the tty at port `tty(1)`.

Offset:
`0x00000005`

Valid baud rates:
`300`, `600`, `1200`, `2400`, `4800`, `9600`, `19200` Bit/sec

Default value:
`9600`

`bootfile`

20 bytes containing the name of the file to boot if the commands `auto` or `boot` (without parameters) are entered. The name must already contain the device driver and number.

Offset:
`0x00000006`

Default value:
`bootfile = dkncr(0,0,10)sash`

`bootmode`

This byte contains the boot mode of the system, set either by the user or the power-on diagnostics [see `prom(8)`].

Offset:
`0x0000001a`

Valid bootmodes:

d, m, c, e, w, r, i

Default value:

m

console

This byte contains information on the consoles which will be enabled automatically at power-on.

Offset:

0x0000001b

Valid console types are:

l The local console only will be enabled. This can be grafic(), tty552(0) or tty(0), depending on the graphical configuration of the system. l is the default console type.

r The local console and tty(1) or tty552(1) (serial port to a host) will be enabled.

a All existing consoles will be enabled.

state state maintains the current validity of the time-of-day clock and the non-volatile RAM.

Offset:

0x0000001c

failcode

failcode is used by the power-on diagnostics to save a failure code for use by service engineers.

Offset:

0x0000001d

cpuid cpuid is the logical slot number for multi-CPUs indicating which slot they are in.

Offset:

0x0000001e

ponmask

The power-on tests store their diagnostic results in this word. Each test group has its own bit within this word; if the corresponding bit is set, an error has occurred. The power-on tests themselves use these bits to decide whether or not to skip the test.

Offset:

0x00000020

resetepc

Location to store epc (exception program counter) in non-volatile RAM.

Offset:

0x00000024

resetra

Location to store ra (return address) in non-volatile RAM.

Offset:

0x00000028

chksum

The firmware decides whether or not the values within non-volatile RAM are valid using this byte. The checksum is built by adding up all NVRAM-contents modulo 0x100 (besides the checksum), computing the 2's complement of the result and storing this value in the byte chksum. This is done every time the NVRAM contents change, via the command setenv. If the NVRAM is valid, the sum of all NVRAM contents modulo 0x100 (including chksum) must be zero.

Offset:

0x00000030

Teleservice, Baudrate

The baud rate is stored in these two bytes. This 2-byte word is "little-endian" (Intel notation).

Offset:

0x0000001c

Valid baud rates:

300, 600, 1200, 2400, 4800, 9600, 19200 Bit/sec

Teleservice, Bit/Character

One byte, containing the character width for the transmitted character.

Offset:

0x00000033

Valid values:

7, 8 Bit/char

Teleservice, Flag-Bits

D7							D0
TSa	TSe	res	res	res	STOP	PAR	PAR

TSa: Teleservice-active-Bit

- 0: Teleservice not active (default)
- 1: Teleservice active

TSe: Enable-Teleservice-Bit

- 0: Predefined V24 S-signals will not be set.
- 1: Predefined V24 S-signals will be set. The Teleservice channel will be initialized (default).

Before booting, the option of activating Teleservice is offered.

STOP: Number of Stop-Bits

- 0: 1 Stop-Bit (default)
- 1: 2 Stop-Bits

PAR: Number of parity check bits

- 00: No parity (default)
- 01: Odd parity
- 10: Even parity
- 11: Invalid

Offset: 0x00000034

Teleservice, V24 Handshake Signals

The two bytes TS HS_0 and TS HS_1 store the mask for all modem signals used. These bytes are used by the firmware to inform the driver which signals should be observed at communication time. The contents of the byte TS HS_1 are not yet defined. The byte, therefore, may not be observed. Contents of the byte TS HS_0:

D7							D0
S1	S2	S4	M3	M4	M1	M2	M5

0: Signal must not be observed

1: Signal must be observed

Offset: 0x00000035

DUART 2681 Channel A

This byte stores the mask for all handshake signals used for the Duart 2681, channel A. Contents of the byte HS:

D7							D0
S1	S2	res	res	res	M1	res	res

0: Signal must not be observed

1: Signal must be observed

Offset: 0x00000037

Asynchronous Communication Element 16C552, Handshake Signals

A byte is reserved in NVRAM for each of the two serial channels of the ACE 16C552. Information about the important interface signals are stored within this byte. Contents of the bytes 552_0 HS and 552_1 HS:

D7							D0
S1	S2	res	M3	res	M1	M2	M5

0: Signal must not be observed

1: Signal must be observed

Offset: 0x00000038 (byte 552_

Offset: 0x00000039 (byte 552_

console channel

The firmware computes the actual console channel at initialization time of the I/O system. The information in this byte is only valid if exactly one bit is set. The order of the test is:

1. graphics controller
2. S2681 channel 0
3. ACE 16C552 channel 0

Contents of the byte console:

| | | | | | | |

GRAFIC: A graphics controller

S2681: S2681 channel 0

16C552: ACE 16C552 channel 0

MEM: Memory

Offset: 0x0000003a

grafic-controller
: The firmware computes the actual console channel at initialization time of the I/O system. Graphics controllers are not currently supported by the firmware.

graf typ
: The information in this byte is an index within a list of known graphics controllers: Not used for the moment.

Offset:
0x0000003b

GrafRes1, GrafRes2
: The bytes GrafRes1 and GrafRes2 are reserved.

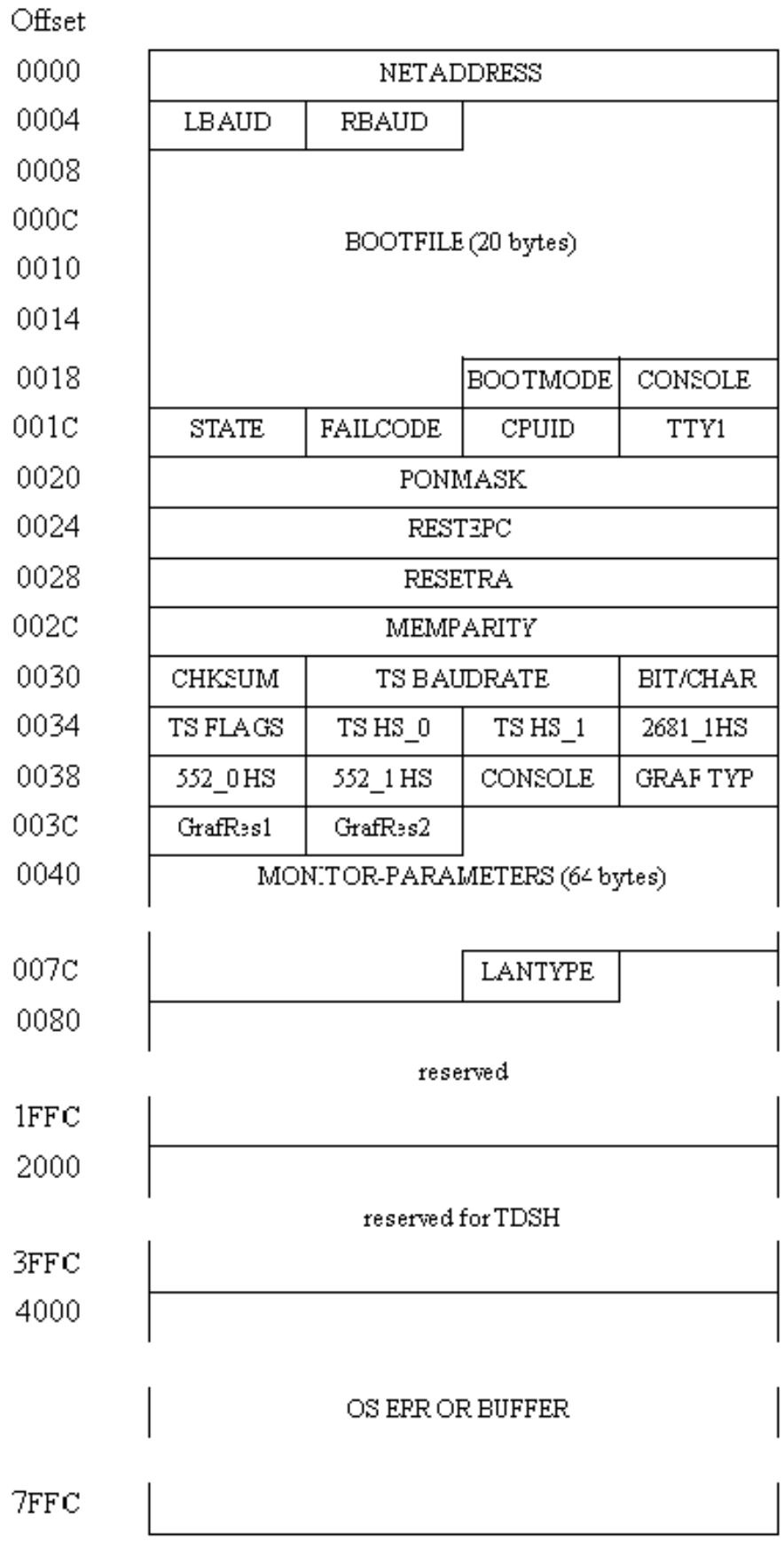
Offset:
0x0000003c

grafic monitor parameters
: The monitor parameters with which the graphics console was initialized are stored within this structure. There is a separate checksum for the data within this structure, consisting of two bytes at the top of the structure: Not used at present.

Offset:
0x0000003e

Locations of contents in non-volatile RAM

Base address of the non-volatile RAM: 0x1c080000


Offset
0000
NETADDRESS
0004
LBAUD
RBAUD
0008
000C
BOOTFILE (20 bytes)
0010
0014
0018
BOOTMODE
CONSOLE
001C
STATE
FAILCODE
CPUID
TTY1
0020
PONMASK
0024
RESTEPC
0028
RESETRA
002C
MEMPARITY
0030
CHKSUM
TS BAUDRATE
BIT/CHAR
0034
TS FLAGS
TS HS_0
TS HS_1
2681_1HS
0038
552_0 HS
552_1 HS
CONSOLE
GRAF TYP
003C
GrafRes1
GrafRes2
0040
MONITOR-PARAMETERS (64 bytes)
007C
LANTYPE
0080
reserved
1FFC
2000
reserved for TDSH
3FFC
4000
OS ERROR BUFFER
7FFC

Default environment variables

This section lists the default environment variables of the PROM monitor and of the SASH using the PROM monitor/SASH command `printenv` (without parameters). Default in this case means that these environment variables are defined at startup time by the firmware (or the SASH). The contents of these environment variables are described in detail in the section **Contents of the non-volatile RAM** above.

An exception is the environment variable `ethernetaddr`. This variable is read-only and stored in the Ethernet PROM on your CPU board. Please also note that not all NVRAM contents belong to the environment (e.g. the NVRAM checksum is not part of the environment).

Additional environment variables are described in the next section.

NOTE:

The contents of the environment variables can differ on your machine, depending on the graphical configuration, hard disk unit number, etc.

```
sash: printenv
netaddr=141.29.51.12□
ethernetaddr=00:00:e4:01:02:0d□
lbaud=9600□
rbaud=9600□
bootfile=dkncr(0,0,10)sash□
bootmode=c□
console=l□
console_dev=tty(0)□
grafic_dev=none□
tty(0)_if=S1(108)/S2(105)/M5(109)□
tty552(0)_if=S1(108)/S2(105)/M5(109)□
tty552(1)_if=S1(108)/S2(105)/M5(109)□
cpuid=0□
resetepc=0xbfc013c4□
resetra=0xbfc01378□
version=2.0011□
lantype=ethernet□
TS_baud=2400□
TS_bit/char=8□
TS_parity=no□
TS_stopbits=1□
TS_if=S1(108)/S2(105)/S4(111)/M1(107)/M2(106)/M5(109)□
TS_flags=tse
```

Additional environment variables

This section describes additional environment variables, i.e. variables not defined by the firmware or the SASH at startup time.

`path` With the variable `path`, input of file names is more flexible and less erroneous. Each command that has a file name as parameter(s) checks first whether the given file can be found. If so, the given file name is selected for operation. If not, `$PATH` is prepended to the file name and the result is checked again to see whether or not it can be found.

`debug` With this variable set to a certain value, diagnostic messages about various functions and commands are displayed during execution. To get such diagnostic messages, the PROM/SASH must have been compiled with the `XPR_DEBUG` compiler flag on.

`debug` can have the following values:

Value	Description

0x0001	Serial line protocol
0x0002	Remote debug protocol
0x0004	print oob (out-of-band) protocol input
0x0008	debug bfs protocol
0x0010	LANCOP (Local Area Network Coprocessor) driver
0x0020	SCSI driver
0x0040	memory related events
0x0080	tod (time-of-day) related
0x0100	sload (serial load)
0x0200	extra
0x0400	floppy driver (SASH only)
0x0800	TAR filesystem driver (SASH only)

Any combination of the flags mentioned above is permitted.

`Other variables`
: Users are free to create their own variables; all existing variables will be passed as environment variables to standalone programs invoked by the PROM monitor or SASH. The only restriction is the limited size of the environment.

SEE ALSO

nvram(8), prom(8), sash(8).

System Administrator's Reference Manual

olr(7)

NAME

`olr` - Online Replacement

DESCRIPTION

In order to improve the availability of a system, defective hardware components can be replaced at runtime without interrupting unaffected hardware, system and application software.

OLR is currently supported for SCSI disk drives, cardcage power supplies, SCSI power supplies, tape drives, CD-ROM drives and fan-out units.

Please refer to the operating manual for your RM system, which tells you which components you can replace yourself or which should be replaced by the Service department.

For further information, please refer to the following manuals:

- *Hardware Configuration with Config under SINIX/windows*, chap. *Exchanging devices while the system is active*
- *System Administration and Hardware Configuration Using the SYSADM Interface*, chap. *Configure the system*, sect. *Exchanging devices while the system is active*
- *Virtual Disks*

SEE ALSO

`act_dev`(1M), `deact_dev`(1M),
only RM400: `act_ctlr`(1M), `deact_ctlr`(1M),
only RM600: `reset_lar`(1M), `set_lar`(1M),
`dktab`(4), `lar`(7), `autoconf`(8), `changedev`(8), `showconf`(8).

System Administrator's Reference Manual

osm(7)

NAME

`osm` - Operating System Messages (OSM)

DESCRIPTION

The kernel and drivers store their messages in the `/dev/osm` console ring buffer. These can be unstructured texts as they appear on the console or structured CMN kernel and DRV driver messages. As of Reliant UNIX V5.44, the kernel and driver messages are no longer duplicated in `/dev/log`. The console ring buffer is 10,000 bytes large and should primarily record the messages that arise before Logging V3.0 or the `syslogd` daemon starts. It is only if more than 10,000 bytes of messages arise before logging starts – depending on the hardware configuration – that messages may be lost through the cyclical overwriting of the console ring buffer.

The `syslogd` daemon is started by the `/etc/rc2.d/S06log3` script when multiuser mode is initiated and begins immediately to read and log `/dev/osm2`, i.e. all messages written to `/dev/osm` from this time on are logged by the `syslogd` daemon. All messages written previously to `/dev/osm` have been collected in the main memory by the kernel and sent to the daemon with `/dev/log` when the `syslogd` daemon was started. Almost the entire main memory is available for storing messages, which means that no messages can be lost.

The `osmd` daemon is started [see `osmd`(1M)] by the `/etc/rc2.d/S09osm` script [see `S09osm`(1M)] when multiuser mode is initiated. The daemon then reads `/dev/osm` and creates log files. The console ring buffer is read from the beginning and further messages are awaited.

The `osmd` daemon creates a cycle of three files, by default, under `/etc/.osm`[*.MM-DD-hh:mm*], whereby the maximum size of each file is 5 MB. The messages are always written to the file without the suffix. If the maximum size of this file will be exceeded by the next entry, the file is assigned a suffix made up of date and time and a new file is created without the suffix. The messages are written to this new file. If the number of files in the cycle will consequently be exceeded, the oldest files are deleted (generally one file unless the number of files belonging to the cycle has been reduced). The defaults for the `osmd` daemon can be overridden by values from the `/etc/default/osm.conf` configuration file [see `osm.conf`(4)]. The signal 1 (`SIGHUP`) must be sent to the `osmd` daemon to enable it to load the modified content of the `/etc/default/osm.conf` configuration file.

	`osmd` defaults	Configuration file `/etc/default/osm.conf`	Range
File name	`/etc/.osm`	`PROTOCOL=/var/adm/log/osm`	Arbitrary
No. cycles	3	`MAX=3`	2 – 100
Maximum size	5 MB	`SIZE=5120000`	10 KB – `ulimit` value

The configuration file supplied ensures that the root file system can no longer overflow. The settings in the configuration file are not overwritten with an update installation.

Invoke the following command for a chronological listing of the protocol for all files in the cycle:

```
cat `ls -tr /var/adm/log/osm*` | pg
```

FILES

`/dev/osm`
: console ring buffer for kernel and driver messages

`/etc/default/osm.conf`
: configuration file for the `osmd` daemon

`/etc/default/osm.filter`
: filter file

`/etc/.osm`[`.`*MM*`-`*DD*`-`*hh*`:`*mm*]
log files

`/var/adm/log/osm`[`.`*MM*`-`*DD*`-`*hh*`:`*mm*]
log files

SEE ALSO

`osm`(1M), `osmd`(1M), `syslogd`(1M), `osm.conf`(4), `log`(7).

System Administrator's Reference Manual

pckt(7)

NAME

`pckt` - STREAMS Packet Mode module

DESCRIPTION

`pckt` is a STREAMS module that may be used with a pseudo terminal to 'packetize' certain messages. The `pckt` module should be pushed [see `I_PUSH`, in `streamio(7)`] onto the master side of a pseudo terminal.

Packetizing is performed by prefixing a message with an `M_PROTO` message. The original message type is stored in the 4 byte data portion of the `M_PROTO` message.

On the read side, only the `M_PROTO`, `M_PCPROTO`, `M_STOP`, `M_START`, `M_STOPI`, `M_STARTI`, `M_IOCTL`, `M_DATA`, `M_FLUSH`, and `M_READ` messages are packetized. All other message types are passed upstream unmodified.

Since all unread state information is held in the master's stream head read queue, flushing of this queue is disabled.

On the write side, all messages are sent down unmodified.

With this module in place, all reads from the master side of the pseudo terminal should be performed with the `getmsg(2)` or `getpmsg(2)` system call. The control part of the message contains the message type. The data part contains the actual data associated with that message type. The onus is on the application to separate the data into its component parts.

SEE ALSO

`crash(1M)`, `getmsg(2)`, `ioctl(2)`, `ldterm(7)`, `ptem(7)`, `streamio(7)`, `termio(7)`.

Programmer's Guide: STREAMS

System Administrator's Reference Manual

ports(7)

NAME

`ports` - 5 line asynchronous communications interface STREAMSdriver

DESCRIPTION

`ports` is a STREAMS-based driver that supports a five line asynchronous interface. Each device supports 4 RS232 lines and one parallel Centronics interface.

When used in conjunction with the STREAMS line discipline module, `ldterm`, behavior on all lines is as described in `termio(7)`.

FILES

`/dev/term/??`
: serial interface

`/dev/lp?`
: parallel interface

SEE ALSO

`crash(1M)`, `ldterm(7)`, `termio(7)`.

Programmer's Guide: STREAMS

System Administrator's Reference Manual

ppa(7)

NAME

ppa - PPA streams module

DESCRIPTION

ppa is the module that handles the asynchronous line mode of ppp(7) . This includes the calculation of the frame check sequence and the mapping of control characters.

The ppa module is pushed onto a stream above a serial line driver. When receiving packets, it accepts a sequence of streams messages and processes them. It passes a single stream message up for each full packet received. On output it adds the frame start and stop sequences and passes the packet down.

The ppa module is controlled by a series of ioctl's. Each ioctl(2) is passed the following ppioctl union.

```
typedef unsigned long   ppp_char_map;
long    ppi_sap;
long    ppi_indx;
long    ppi_flags;
long    ppi_mtu;
long    ppi_timeout;
ppa_cnt_t ppi_cnt;
ppp_char_map ppi_charmap;
struct  _ppi_setstropts {
        long    _pps_sap;
        long    _pps_flags;
} ppi_stropts;
```

The following options are permitted.

PPA_GETSTATS
: This command returns a ppa_cnt_t structure which contains statistics about ppa. These include the number of packets dropped and the number of raw characters transmitted. This information is used by the LQM protocol.

PPA_SETRCVMASK, PPA_SETSNDMASK
: These commands set the asynchronous control mask for reception and transmission respectively.

PPA_SETMTU, PPA_GETMTU
: These commands set/get the maximum transfer unit length for transmission.

PPA_SETMAXIPKT, PPA_GETMAXIPKT
: These commands set/get the maximum receivable packet size.

SEE ALSO

ppp(7) .

System Administrator's Reference Manual

ppp(7)

NAME

ppp - PPP streams module

DESCRIPTION

ppp is the driver that handles the demultiplexing of incoming PPP packets based on the protocol id included in the header of a PPP packet. The packet is then processed by a protocol-specific function. If no function is defined for the stream, the packet is passed up transparently. Most of the work of this driver is in handling the DLPI interface needed for an IP packet provider.

On output, ppp adds the PPP protocol header and passes the packet down the stream.

When a process first opens the driver, a PPP device block is allocated to the process, this is associated with the process until the control stream is closed. A PPP device block has one lower stream to which a serial line controller stream can be I_LINKed. When a process opens the driver a second time, a second upper stream is created to the same lower stream. Input and output processing are upper stream dependent. They can be changed by the use of ioctls.

IOCTLS

The operation of the multiplexer is controlled by a series of ioctls which pass the following ppioctl union.

```
typedef unsigned long   ppp_char_map;
long    ppi_sap;
long    ppi_indx;
long    ppi_flags;
long    ppi_mtu;
long    ppi_timeout;
ppa_cnt_t ppi_cnt;
ppp_char_map ppi_charmap;
struct  _ppi_setstropts {
        long    _pps_sap;
        long    _pps_flags;
} ppi_stropts;
```

The following options are permitted.

PPP_GETDEV
: This command returns the device number of connection in ppi_indx. It is used to make a name for IP connections.

PPP_ADDSAP
: This command adds a SAP (PPP protocol id used on reception to demultiplex incoming packets) supplied in ppi_sap to the current stream. If the SAP is already in use on the current device then an error results.

PPP_DELSAP
: This command deletes a given SAP from the current stream. (This is not currently used by PPD and is not guaranteed to work).

PPP_SETSAPFLAGS
: This sets the flags for the supplied SAP. The only flag currently recognized is the PPP_SAP_VALID flag. If this flag is set, the SAP is used for demultiplexing, if it is not set, the SAP is ignored.

PPP_GETSAPFLAGS
: This command returns the current SAP's flag word.

PPP_SETQUEFLAGS
: This command sets the current streams flags. The valid flags are given below:

```
PPP_DATACHAN            0x2
PPP_OUT_ENABLE          0x4
```

```
PPP_OUT_SAP_SET          0x8
PPP_PFC_COMPRESS         0x10
PPP_ACFC_COMPRESS        0x20
PPP_VJ_COMPRESS          0x40
PPP_WAIT_OUTPUT          0x80
PPP_OUT_SAP_ENABLE       0x8000
```

The PPP_DATACHAN flag informs the system that this stream will carry data and that netstat(1M) statistics for the connection should be kept.

The PPP_OUT_ENABLE flag enables transmission on this stream. If this bit is not set, then no output will be sent.

The PPP_OUT_SAP_SET flag is used to specify the SAP that is to be used when a packet is transmitted on the stream. This only needs to be set if the stream has more than one SAP associated with it. This bit is only checked if the PPP_OUT_SAP_ENABLE is set. If this bit is not set then the current value of PPP_OUT_SAP_SET is used.

The PPP_PFC_COMPRESS and PPP_ACFC_COMPRESS flags are used to enable the compression of PPP frames on compression.

The PPP_VJ_COMPRESS flag enables the transmission of VJ compressed TCP/IP frames. This bit is only valid for streams with an IP protocol outbound SAP.

The PPP_WAIT_OUTPUT flag is used by the –W flag of PPD. It enables a special short circuit of IP frames to be set up. It is not for normal use.

PPP_GETQUEFLAGS
: This command returns the current set of stream flags as described above.

PPP_SETDEVFUNC
: This command sets the function to be used when data is received. The currently defined values are:

```
PPP_PROTOFUNC    1
PPP_IPDATAFUNC   2
PPP_DATAFUNC     3
```

If no function is specified, the packet is passed up raw. A value of –1 for the function resets the connection to the default state.

The PPP_PROTOFUNC is used by PPD for most protocols. The inbound frame is split into two parts: a control message with the length of a long, which contains the protocol id of the packet received, and the data part which contains the data part of the packet.

The PPP_IPDATAFUNC is used by IP to encapsulate the data in a DLPI message. PPP_DATAFUNC also encapsulates the data in a DLPI message. The difference is that PPP_IPDATAFUNC also handles VJ compressed packets.

PPP_SETTIMER
: This option sets the length of time (in seconds) for which the connection can be idle before it is closed. This is achieved by sending a HUP message up the controlling stream.

PPP_SETCTRL
: This command is used to set the controlling stream of the lower device. This is normally the first stream opened but in some circumstances this must be changed. The controlling stream is special in that if it is closed, all other streams associated with the current device are disconnected from the lower stream. The controlling stream also receives all packets with unknown SAPS, and any hangup messages from the lower stream.

FILES

/dev/ppp

SEE ALSO

ppd(1M) .

System Administrator's Reference Manual

prf(7)

NAME

`prf` - operating system profiler

DESCRIPTION

The special file `/dev/prf` provides access to activity information in the operating system. Writing the file loads the measurement facility with text addresses to be monitored. Reading the file returns these addresses and a set of counters indicative of activity between adjacent text addresses.

The recording mechanism is driven by the system clock and samples the program counter at line frequency. Samples that catch the operating system are matched against the stored text addresses and increment corresponding counters for later processing.

The file `/dev/prf` is a pseudo device with no associated hardware.

FILES

`/dev/prf`

SEE ALSO

`profiler`(1M).

System Administrator's Reference Manual

ptem(7)

NAME

ptem - STREAMS Pseudo Terminal Emulation module

DESCRIPTION

ptem is a STREAMS module that when used in conjunction with a line discipline and pseudo terminal driver emulates a terminal.

The ptem module must be pushed [see I_PUSH, in streamio(7)] onto the slave side of a pseudo terminal, before the ldterm module is pushed.

On the write side, the TCSETA, TCSETAF, TCSETAW, TCGETA, TCSETS, TCSETSW, TCSETSF, TCGETS, TCSBRK, JWINSIZE, TIOCGWINSZ, and TIOCSWINSZ termio ioctl messages are processed and acknowledged. A hang up (i.e. stty 0) is converted to a zero length M_DATA message and passed downstream. termio cflags and window row and column information are stored locally one per stream. M_DELAY messages are discarded. All other messages are passed downstream unmodified.

On the read side all messages are passed upstream unmodified with the following exceptions. All M_READ and M_DELAY messages are freed in both directions. An ioctl TCSBRK is converted to an M_BREAK message and passed upstream and an acknowledgement is returned downstream. An ioctl TIOCSIGNAL is converted into an M_PCSIG message, and passed upstream and an acknowledgement is returned downstream. Finally an ioctl TIOCREMOTE is converted into an M_CTL message, acknowledged, and passed upstream.

FILES

<sys/ptem.h>

SEE ALSO

stty(1), crash(1M), ioctl(2), ldterm(7), pckt(7), streamio(7), termio(7).

Programmer's Guide: STREAMS

System Administrator's Reference Manual

ptf(7)
RM400 only

NAME

`ptf` - power failure and temperature checking pseudo device

DESCRIPTION

The pseudo-device `/dev/at/ptf` gives the user access to the current state of the power and temperature sensors. The only permissible operations on this device are `open(2)` for reading, `read(2)` , and *close*(2) with the restriction that only one `open(2)` can be active at any time. A request to open the device while it is already opened, will fail and `errno` will be set to `EBUSY`. If any attempt is made to open the device for writing, the attempt will fail and `errno` will be set to `ENODEV`.

Exactly four ASCII characters are available to be read: three letters and a carriage return (ASCII □012) character.

The first character describes the state of the power fail sensor: an uppercase "P" is returned when the main power is OK, otherwise a lowercase "p" is returned.

The second character describes the state of the temperature sensor inside the cabinet: an uppercase "T" is returned if the temperature is below a critical limit (which is hardware-dependent), otherwise a lowercase "t" is returned.

If the fan fails, a lowercase "f" is output, otherwise an uppercase "F" is returned. This is dependent on the system type.

If the main power fails or if the temperature exceeds the critical limit, the driver sends a signal (`SIGPWR`) to the `init` process.

EXAMPLE

```
% cat /dev/at/ptf
PT
```

FILES

`/dev/at/ptf`
`/sbin/powerfail.sh`

SEE ALSO

`init(1M)` .

System Administrator's Reference Manual

pw(7)

NAME

`pw` - post & wait driver

SYNOPSIS

```
#include <sys/pw.h>
```

DESCRIPTION

This pseudo driver has been developed especially for database purposes and combines process synchronization and a timeout mechanism of a fine resolution. Up to four independent process groups can currently use this driver. Each of these groups uses one of the pseudo devices:

`/dev/pw/[0-3]`

This allows four independent applications to use this driver. The use of this interface is controlled by `ioctl`. The following `ioctl` functions are supported:

`PW_TIME`

This simple function allows a process to sleep for the number of ticks given in the argument.

Example: `ioctl(fd, PW_TIME, 2*HZ)`

will sleep for 2 seconds.

`PW_POST`

Wakes up a process with a logical ID for this application. Only the process which is waiting with `PW_WAIT` on the same ID will be woke up. The argument is a logical ID ranging between 0 and 255.

Example: `ioctl(fd, PW_POST, l_id)`

wakes up the process with logical ID "l_id".

`PW_WAIT`

This function waits either for a specified amount of clock ticks or to be woken up by another process with a specific logical ID. If this process is woken up by another process using the same logical ID, `ioctl` returns `PW_POSTED`. Otherwise, it returns `PW_TIMEDOUT` to indicate that it has waited for the specified number of clock ticks.

The argument for `ioctl` has the following structure: The three most significant bytes represent the number of clock ticks to wait. The least significant byte is the logical ID for this application in the same range as described above.

Example: `ioctl(fd, PW_WAIT, (100 << 8) | l_id)`

waits for a maximum of 100 ticks or until it is woke up by another process with ID "l_id".

`PW_WAIT_NT`

This function is similar to the PW_WAIT function. The difference is that no timeout has to be specified. The calling process will go to sleep and has to be waken up either by a PW_POST ioctl or a signal. If this process is woken up by another process using the same logical ID, `ioctl` returns `PW_POSTED`. This function should be used when the application does not need a timeout handling.

The argument for `ioctl` has the following structure: The least significant byte is the logical ID for this application in the same range as described above. The three most significant bytes are not evaluated.

`PW_TIME_OFFSET`

Copies the offset from "lbolt", to the address specified in arg. In conjunction with `mmap` the user has direct access to "lbolt" to get read-only access to the tick counter.

Example:

```
addr=mmap(NULL, PAGESIZE, PROT_READ, MAP_SHARED, fd, 0);
ioctl(fd, PW_TIME_OFFSET, &offset);
p_lbolt = (u_long*) (addr+offset);
actual = *p_lbolt;
```

The current value of "lbolt" can be accessed with this sequence after the number of specified clock ticks has expired.

The use of `PW_TIME_OFFSET` will fail, if the file descriptor returned from `open` has not been opened with `O_RDONLY`!

DIAGNOSTICS

If the sleeptime is negative or the logical ID is not valid, a value of −1 is returned and errno is set to `EINVAL`.

System Administrator's Reference Manual

rtty(7)

NAME

`rtty` - remote LAN terminal connection

DESCRIPTION

Remote **TTY** means the connection of a LAN device to a Reliant UNIX host, so that it represents a normal TTY terminal. These `rtty` end devices can be addressed by means of character special device nodes [see `mknod`(1M)]. The `rtty` concept is based on the division of a network connection into different components. These `rtty` components are a

- SAC-driven monitor `rtymon` in the context of the "Service Access Facility" of the Reliant UNIX system and a
- STREAMS multiplexer `rtymux` in the Reliant UNIX operating kernel, through which a character special device node for the application is available. It allows access to the major device number for the application and serves the TTY system call interfaces [see `intro_prm2`(2)].

Connection setup and monitoring of the connection to the LAN device is carried out by `rtymon`. The device port in the LAN is specified in its database (`_pmtab`). In accordance with this description, it establishes a socket connection and links (`I_LINK`) the device nodes of the `rtymux` multiplexer. The connection is monitored, and the `rtty` device node is disconnected by the monitor from the socket connection as they terminate.

The `rtymux` multiplexer transfers the data from the device node to the socket connection of the LAN port, if it is switched by the `rtymon` to the socket connection.

This distribution of tasks necessitates communication between the two `rtty` components. For this reason, the multiplexer with the monitor for each `rtty` end device is connected to the "administration channel" by means of a character special device node.

To meet the requirement for suitable diagnostic help, the multiplexer offers another character special device node, the "diagnostic channel". There is one of these for each connection. It can be opened at any time to obtain status information on the connection. The multiplexer trace can also be activated and deactivated.

DIAGNOSTICS

For diagnostic purposes, the `rtymux` multiplexer has a "diagnostic channel" for each `rtty` end device. This channel can be used to obtain status and diagnostic information with the tools provided. For example, the `probe`(1M) command can be used with the component name <`rtymux`> to obtain all kernel information on the multiplexer. A debugger program `rtydb`(1M) is provided to help experts to simulate and analyze the `rtty` situation. Finally, there is an `rtymux_trace`(1M) command for activating a trace for data exchange on a selected end device.

FILES

`/opt/diag.d/bin`

`/usr/lib/saf`

`/etc/saf/`*tag*`/_pmtab`

SEE ALSO

`diagdispl`(1M), `mknod`(1M), `probe`(1M), `rtyadm`(1M), `rtydb`(1M), `rtymon`(1M), `rtymux_trace`(1M), `sac`(1M), `termadd`(1M), `intro_prm2`(2), `ioctl`(2).

System Administrator's Reference Manual

sad(7)

NAME

`sad` - devices administered by System Configuration

DESCRIPTION

The files in the `/dev/sad` directories (for block devices) are used by System Configuration to access the devices on which it operates.

FILES

`/dev/sad`

SEE ALSO

`sysadm`(1M) .

System Administrator's Reference Manual

sad_strm(7)

NAME

`sad_strm` - STREAMS Administrative Driver

SYNOPSIS

```
#include <sys/types.h>
#include <sys/conf.h>
#include <sys/sad.h>
#include <sys/stropts.h>

int ioctl (fildes, command, arg);
int fildes, command;
```

DESCRIPTION

The STREAMS Administrative Driver provides an interface for applications to perform administrative operations on STREAMS modules and drivers. The interface is provided through `ioctl(2)` commands. Privileged operations may access the `sad` driver via `/dev/sad/admin`. Unprivileged operations may access the `sad` driver via `/dev/sad/user`.

fildes is an open file descriptor that refers to the `sad` driver. *command* determines the control function to be performed as described below. *arg* represents additional information that is needed by this command. The type of *arg* depends upon the command, but it is generally an integer or a pointer to a *command*-specific data structure.

COMMAND FUNCTIONS

The `autopush(1M)` facility allows one to configure a list of modules to be automatically pushed on a stream when a driver is first opened. `autopush` is controlled by the next commands.

`SAD_SAP`

Allows the administrator to configure the `autopush` information for the given device. *arg* points to a `strapush` structure which contains the following members:

```
uint   sap_cmd;
long   sap_major;
long   sap_minor;
long   sap_lastminor;
long   sap_npush;
uint   sap_list[MAXAPUSH] [FMNAMESZ + 1];
```

The `sap_cmd` field indicates the type of configuration being done. It may take on one of the following values:

`SAP_ONE`
: Configure one minor device of a driver.

`SAP_RANGE`
: Configure a range of minor devices of a driver.

`SAP_ALL`
: Configure all minor devices of a driver.

`SAP_CLEAR`
: Undo configuration information for a driver.

The `sap_major` field is the major device number of the device to be configured. The `sap_minor` field is the minor device number of the device to be configured. The `sap_lastminor` field is used only with the `SAP_RANGE` command, with which a range of minor devices between and including `sap_minor`

and sap_lastminor are to be configured. The minor fields have no meaning for the SAP_ALL command. The sap_npush field indicates the number of modules to be automatically pushed when the device is opened. It must be less than or equal to MAXAPUSH, defined in sad.h. It must also be less than or equal to NSTRPUSH, the maximum number of modules that can be pushed on a stream, defined in the kernel master file. The field sap_list is an array of module names to be pushed in the order in which they appear in the list.

When using the SAP_CLEAR command, the user sets only sap_major and sap_minor. This will undo the configuration information for any of the other commands. If a previous entry was configured as SAP_ALL, sap_minor should be set to zero. If a previous entry was configured as SAP_RANGE, sap_minor should be set to the lowest minor device number in the range configured.

On failure, errno is set to the following value:

EFAULT
arg points outside the allocated address space.

EINVAL
The major device number is invalid, the number of modules is invalid, or the list of module names is invalid.

ENOSTR
The major device number does not represent a STREAMS driver.

EEXIST
The major-minor device pair is already configured.

ERANGE
The command is SAP_RANGE and sap_lastminor is not greater than sap_minor, or the command is SAP_CLEAR and sap_minor is not equal to the first minor in the range.

ENODEV
The command is SAP_CLEAR and the device is not configured for autopush.

ENOSR
An internal autopush data structure cannot be allocated.

SAD_GAP
Allows any user to query the sad driver to get the autopush configuration information for a given device. *arg* points to a strapush structure as described in the previous command.

The user should set the sap_major and sap_minor fields of the strapush structure to the major and minor device numbers, respectively, of the device in question. On return, the strapush structure will be filled in with the entire information used to configure the device. Unused entries in the module list will be zero-filled.

On failure, errno is set to one of the following values:

EFAULT
arg points outside the allocated address space.

EINVAL
The major device number is invalid.

ENOSTR
The major device number does not represent a STREAMS driver.

ENODEV
The device is not configured for autopush.

SAD_VML
Allows any user to validate a list of modules (i.e. to see if they are installed on the system.) *arg* is a pointer to a str_list structure with the following members:

```
int                 sl_nmods;
struct str_mlist    *sl_modlist;
```

The str_mlist structure has the following member:

```
char                    l_name[FMNAMESZ+1];
```

`sl_nmods` indicates the number of entries the user has allocated in the array and `sl_modlist` points to the array of module names. The return value is 0 if the list is valid, 1 if the list contains an invalid module name, or –1 on failure. On failure, `errno` is set to one of the following values:

`EFAULT`
: *arg* points outside the allocated address space.

`EINVAL`
: The `sl_nmods` field of the `str_list` structure is less than or equal to zero.

DIAGNOSTICS

Unless specified otherwise above, the return value from `ioctl` is 0 upon success and –1 upon failure with `errno` set as indicated.

SEE ALSO

`intro_prm2`(2), `ioctl`(2), `open`(2).

Programmer's Guide: STREAMS

System Administrator's Reference Manual

sdisk(7)
RM600 only

NAME

`sdisk` - SCSI disk interface on an RM600

SYNOPSIS

`/dev/ios` *b*`/[ r ]sdisk`*ccusp*

`/dev/iof0/ [ r ]sdisk.`*cc*`.`*nnnnn*`sp`

DESCRIPTION

`sdisk` enables access to SCSI disk devices (magnetic disks and optical disks), which are connected by means of a SCSI host adapter or fiber channel controller. SCSI disk devices are accessed via the special files in the directory `/dev/ios0`. The naming convention for all SCSI disk devices is as follows:

`/dev/ios`*b*`/[r]sdisk`*ccusp*

b Bus number: always 0.

`r` Designates the raw (character) device. If `r` is not specified, a block device is used.

cc The logical controller number. The controller number is any 2-character decimal number between 00 and 99. The number indicates which SCSI bus the device is connected to on which host adapter. Physical controllers are assigned to the controller numbers in `/etc/controller`.

u The SCSI disk drive number (address). This address can be any number between 0 and 15. For values 10 to 15 the hex representation a to f will be used. It is important to ensure that the device address is not the same as the SCSI ID of the host adapter.

p The partition number (any number between 0 and 15). This number can be a single or dual character decimal number.

The following names can be used for access in the case of fiber channel disks (FC):

`/dev/iof0/[r]sdisk.`*cc*`.`*nnnnn*`sp`

The addressing is the same as in the familiar SCSI world. The *cc* and *p* parameters have the same meaning as in the case of SCSI disks.

nnnnn is a value which is generated from the universally unique name for FC devices.

This name will continue to be used in error messages and statistics.

In addition, it is possible to address a disk by means of the serial number:

`/devices/[r]`*serial_number*

`/devices/[r]`*serial_number*`.priv`

Disks can be addressed by serial numbers in this way independently of their position; this simplifies administration, for example in the case of observe or multihost configurations. This addressing is used by Vdisk Lite and applies for SCSI and FC disks.

In contrast to earlier addressing, there are no longer16 partitions here: `/devices/[r]`*serial_number* refers to the usable data area on a disk and `/devices/[r]`*serial_number*`.priv` refers to the area reserved for `statesave` disks.

DEVICE(CLASS)NUMBER FOR SCSI DISKS

The major-minor device number can be determined via `/sbin/autoconf -d` *device_name*. The meaning of the number returned is described below.

The major-minor device number is a 32-bit number in the following form:

BBBMMMMM MMMMMMcc ccccc`0`*dd duuupppp*

B Bus number: always 0.

M The major device number.

c The logical controller number of the host adapter. This can be any number between 00 and 99.

0 Not used, 0.

d The SCSI disk drive number.

u Not used (always 0).

p The partition number.

DEVICE(CLASS)NUMBER FOR FIBER CHANNEL DISKS

In contrast to SCSI disks, a different device number is used here. In addition, these disks do not have a fixed assignment of bits for the controller numbers and the SCSI ID; the device number is assigned dynamically and is only known to the device file system (`devfs`).

ERROR MESSAGES

The error messages for the SCSI disk drives are displayed in the following format:

```
sdisk123s4: <cmd> Test Unit Ready
            <status> 05240200 (hard 1/1) Illegal Request - CDB Illegal Field (MP2
5)
            Extended Sense Bytes 18-21 for lpb at ff1b9ce0 * 00000000
```

or

```
sdisk123s4: <cmd> Write 8 <seek> [4210:8:0] <dkblk> 32 <pblk> 50528
            <status> 01140200 (hard 1/1) Recovered Error - No Record Found (MP25)
            Extended Sense Bytes 18-21 for lpb at ff18b440 * a3010002
```

The second error message is described below:

`sdisk123s4:`
Error on the SCSI drive. In this case, the reference is to the disk drive on the logical controller 12, drive 3, partition 4.

`<cmd>`
Error in the SCSI command. In this case, an attempt was made to carry out a write with eight physical sectors.

`<seek>`
The start address of the write described by [*cylinder*:*head*:*sector*]. Note that the start address is determined in one of two ways. If the `extended sense` information bytes of the SCSI `request sense` command are valid, then this value is decoded into cylinder, head and sector values. Otherwise, the start address specified by the SCSI `write` command is decoded. In either case, the start address does not take into account any remapping that might have been done previously using `format`. So remapping is not based on the start address.

`<dkblk>`
The 512-byte disk block start number (offset from the current partition). In this case, the eight-sector write started at block 32 in partition 4.

`<pblk>`
The physical disk block number. Note that the physical block number can be established in one of two ways. If the `extended sense` information bytes of the SCSI `request sense` command are valid, then this value is used. Otherwise, the start address specified (in this case by the SCSI `write` command) is used. Use this value to specify a start address in `format`.

`<status>` *KKAASSCC*
The IOS composite status code. The composite status code has the following format:

KK The SCSI sense key value.

AA The SCSI additional sense key value.

SS The SCSI status returned to the SCSI host adapter.

CC The SCSI host adapter error code.

`(hard 1/1)`
: The error type, followed by the retry count. There are two error types: hard errors and soft errors. An attempt can be made to recover soft errors (retry). Unrecoverable soft errors and errors that should not be retried are classified as hard errors. The retry count is expressed as the number of retries divided by the maximum number of retries allowed. In the case of hard errors, the value is always 1/1.

`Recovered Error`
: The decoded SCSI sense key value message.

`No Record Found`
: The decoded SCSI additional sense key value message.

`(MP25)`
: The disk drive type.

`Extended Sense Bytes`
: The byte address of the saved `extended sense` data which is displayed in the lines after this message. In this case, bytes 18 to 21 of the MP25 disk's `extended sense` data were saved after sending the SCSI `request sense` command following the error on the SCSI `write` command.

`lpb at`
: The address of the local parameter block for the SCSI `write` command issued to the disk.

`* a3010002`
: The `extended sense` data. The amount of data displayed depends on the number of saved `extended sense` bytes specified for the disk type's profile in `/etc/conf/pack.d/space.c`. Consult the manufacturer's disk manual (`request sense` SCSI command) to decode these bytes.

SCSI DISK CONFIGURATION

The microprogram correction levels and the date code of all SCSI controllers are checked at RM600 boot time. A disk is only configured, however, when the device has been opened. During system load, a disk is identified solely and uniquely on the basis of its inquiry name, as output by the SCSI `inquiry` command. Once a disk has been opened, and SCSI information has been loaded, the disk is configured. The disk configuration can be checked at any time using the `autoconf` command. A sample `autoconf -l` output is given below:

```
ios0/sdisk001 MP14 -- MP14
ios0/sdisk003 MP12: not configured
ios0/sdisk006 MP14 -- partition not set
iof0/sdisk.15.f2073 HD06 -- HD06
iof0/sdisk.15.3f8eb HD06 -- HD06
```

Two fields follow the disk name: the type name and the partition name. Thus, `ios0/sdisk001` has the type name `MP14` and the partition name `MP14`. A shortened form of the SCSI inquiry name, followed by information about the subsequent configuration is displayed for disks that have not yet been configured. Disks that are partially configured (i.e. disks for which `dkindex` has not been run) are indicated by the message `partition not set`. For further information on configuring disks, see `dkindex(8)` and `dkpsdfinit(8)` .

The SCSI disks, as supplied, use the start/stop option. This means that the drive will spin up as soon as the SCSI command is issued. If the drive is already spinning, the SCSI start command has no effect. Drive spin-up requires a few seconds per drive, and is only noticeable if the drives were previously powered down.

After drive spin-up, the disk format and geometry parameters are loaded using the SCSI `mode sense` command. This occurs after the first `open(2)` . Removable media devices are reconfigured after the first `open(2)` .

SCSI DISK NOTES

The physical sector size is not necessarily 512 bytes. The SCSI disk driver converts the block count into the number of sectors required for a 512-byte transfer. When errors occur, both the physical block size and the system block size are output. The format function expects details of the physical block size, should a user wish to map out a bad block.

REMOVABLE MEDIA DISKS

Removable media disks are locked into the drive as soon as the device is opened and remain locked until the device is closed. This prevents a user from inadvertently removing the media while the disk is still is use. Unfortunately, not all SCSI disk drives support removable media disk locking, so this procedure is not foolproof. The last close(2) unlocks the media for possible removal, but does not spin down the drive. After the media has been released, the drive will be reconfigured the next time it is open(2) ed. Use the autoconf(8) command to verify any configuration changes.

In the case of ROD (Rewritable Optical Disk) drives, two modes can be distinguished when writing: writing with or without preceding deletion. As with optical disks, all partitions extend over the entire medium, the partition number can be used to select the operating mode. When writing to partition 0 to 7, writing without deletion takes place, while when writing to partition 8 to 14 this occurs with deletion. Writing without deletion is only permissible if you are writing to unwritten blocks or blocks that have already been deleted. Otherwise, writing with deletion takes place.

SEE ALSO

devfs(7), vdisklite(7), autoconf(8), dkindex(8), dkpart(8), dkpsdfinit(8), dksetup(8), format(8), setinfo(8).

System Administrator's Reference Manual

sdisk(7)
RM400 only

NAME

`sdisk` - SCSI disk interface on an RM400

SYNOPSIS

`/dev/ios` *b*/[`r`]`sdisk`*cusp*

DESCRIPTION

`sdisk` provides SCSI magnetic and optical disk devices. SCSI disk devices are accessed through the special files in the directory `/dev/ios0`. The naming convention for all SCSI disk devices is as follows (all numbers are specified in decimal):

`/dev/ios`*b*/[`r`]`sdisk`*cusp*

b Bus number (always 0).

`r` Designates the raw (character) device.

c The logical controller number (channel ID). The controller number is any 2-decimal number between 00 and 99. The number indicates to which SCSI bus the device is connected.

u The device's unit number (address). The address may range from 0 to 15. For values 10 to 15 the hex representation a to f will be used.

p Designates the disk slice (partition) (0 to 15).

MAJOR-MINOR DEVICE NUMBER

The major-minor device number may be determined via `/sbin/autoconf -d`*device_name*.

The major-minor number is a 32-bit number formatted as follows:

BBBMMMMM MMMMMMcc ccccdddd duuupppp

B Bus number (always 0).

M The major device number.

c The SCSI host adapter's controller number (channel ID).

d The SCSI device address (0 to 15) on the specified SCSI bus.

u The SCSI Logical Unit Number of the specified SCSI device, if supported. Most devices do not use LUN's, therefore bits 4-6 will usually be 0.

p The partition number (0 to 15).

ERROR DESCRIPTION

The error messages displayed for the IOS disk drives have the following formats:

```
sdisk012s3: <cmd> Test Unit Ready
            <status> 05240200 (hard 1/1) Illegal Request - CDB Illegal Field (HP9
7548)
            Extended Sense Bytes 18-21 for lpb at ff1b9ce0 * 00000000
```

or

```
sdisk012s3: <cmd> Write 8 <seek> [4210:8:0] <dkblk> 32 <pblk> 50528
            <status> 01140200 (hard 1/1) Recovered Error - No Record Found (HP975
48)
            Extended Sense Bytes 18-21 for lpb at ff18b440 * a3010002
```

The second error message will be described below:

`sdisk012s3:`
The IOS disk drive in error. In this case, the disk drive located at SCSI bus 01, drive 2, in partition

3.

`<cmd>`
: The SCSI command in error. In this case a write of eight physical sectors was attempted.

`<seek>`
: The seek address for the write described by [*cylinder*:*head*:*sector*]. Note that the seek address is determined in one of two ways. If the SCSI request sense command extended sense information bytes are valid, then this value is decoded into cylinder, head, and sector values. Otherwise the start address specified in (this case by) the SCSI write command is decoded. In either case, the seek address does not take into account any remapping that might have been done previously by `format`. So remapping is not based upon the seek address.

`<dkblk>`
: The 512-byte disk block start number (offset from the current partition). In this case the eight sector write started at block 32 in partition 3.

`<pblk>`
: The physical disk block number. Note that the physical disk block number is determined in one of two ways. If the SCSI request sense command extended sense information bytes are valid, then this value is used. Otherwise the start address specified in (this case by) the SCSI write command is used.

`<status>` *KKAASSCC*
: The IOS composite status code. The composite status code has the following format:

  KK The SCSI Sense Key value.

  AA The SCSI Additional Sense Key value.

  SS The SCSI status returned to the IOS host adapter.

  CC The IOS host adapter error code.

`(hard 1/1)`
: The error type, followed by the retry count. There are two error types: hard and soft. Soft errors are retried. Unrecoverable soft errors and errors that should not be retried are classified as hard. The retry count is expressed as the number of retrys divided by the maximum retrys allowed. In the case of hard errors, the value is always 1/1.

`Recovered Error`
: The decoded SCSI Sense Key value message.

`No Record Found`
: The decoded SCSI Additional Sense Key value message.

`(HP97548)`
: The disk drive type.

`Extended Sense Bytes`
: The byte address of the saved extended sense bytes displayed in the next lines after this message. In this case, bytes 18 to 21 of the HP97548 disk's extended sense data were saved after sending the SCSI request sense command after the error on the write SCSI command.

`lpb at`
: The kernel address of the local parameter block for the SCSI write command issued to the disk.

`* a3010002`
: The extended sense data. The amount of data displayed depends upon the number of saved extended sense bytes specified for the disk type's disk profile in `/etc/conf/pack.d/space.c`. Consult the manufacturer's disk manual (request sense SCSI command) to decode these bytes.

SCSI DISK CONFIGURATION

SCSI disk devices are identified and initialized during boot time in the SCSI drive probe. A disk is not actually configured however, until it has been opened. During system boot, a disk is identified solely on the basis of its SCSI inquiry name. Once a disk has been opened, and SCSI geometry information has been retrieved, the disk is configured. Disk configuration may be checked at any time via the `autoconf` command. Below is

sample `autoconf -l` output:

```
ios0/sdisk001 HP 97548 -- HP 97548
ios0/sdisk003 CDC 94171: not configured
```

Following the disk name are two fields: the type, and partition names. So, `ios0/sdisk001` has a type name of `HP 97548`, and a partition name of `HP 97548`. Disks which haven't been configured yet display a shortened form of the SCSI inquiry name followed by an unconfigured message.

After drive spin up, the disk format and geometry parameters are retrieved through the mode sense SCSI command. This step occurs after the first `open(2)`. Removable media devices are reconfigured after the first `open(2)`.

SCSI DISK NOTES

The physical sector size is not necessarily 512 bytes. The SCSI disk driver converts the block count into the number of sectors required for a 512 byte transfer. When errors occur both the physical block and the system block size are printed. The formatter expects that the physical block will be provided, should a user wish to map out a bad block.

REMOVABLE MEDIA DISKS

Removable media disks differ from fixed disks in that they are locked into the drive after the first open and remain locked until after the last close. This prevents a user from inadvertently removing the media while the disk is still in use. Unfortunately not all removable media SCSI disk drives support media locking, so this scheme is not foolproof. The last `close(2)` will unlock the media for possible removal, but will not spin the drive down. After the media has been released, subsequent `open(2)` s will cause the drive to re-configure. Use the `autoconf` command to verify any configuration changes.

In the case of ROD (Rewritable Optical Disk) drives, two modes can be distinguished when writing: writing with or without preceding deletion. As with optical disks, all partitions extend over the entire medium, the partition number can be used to select the operating mode. When writing to partition 0 to 7, writing without deletion takes place, while when writing to partition 8 to 14 this occurs with deletion. Writing without deletion is only permissible if you are writing to unwritten blocks or blocks that have already been deleted. Otherwise, writing with deletion takes place.

SEE ALSO

`autoconf(8)`, `dkpart(8)`, `format(8)`, `setinfo(8)`, `upd_channel_id(8)`.

System Administrator's Reference Manual

sduart(7)
RM400 only

NAME

`sduart` - general serial interface

DESCRIPTION

The RM400 contains on board two serial controllers (SCN2681 and VL16C552). Each controller has two physical channels. The channels are connected to seven logical devices. The logical devices are accessed through special files in the `/dev/term` directory.

At the moment, all serial devices controlled by the `sduart` driver automatically push the modules `ldterm(7)` and `ttcompat(7)` (`/etc/ap/cons.ap`).

The following logical devices are supported:

2681 – PORT A

Physical channel:
2681 – port A

Logical device:
`/dev/term/00` or `/dev/constty`

Major device number:
128

Minor device number:
0

This logical device only knows the transmitter and receiver data lines. Normally, it can be used as a console, but any serial device can also be controlled over this channel.

2681 – PORT B

The "2681 – Port B" channel only accepts one link to a logical channel. An opening to other logical channels will be immediately refused (`O_NONBLOCK=1`) or will remain dormant until the existing link to a second logical device breaks (close).

`/dev/term/01`

Physical channel:
2681 – port B (modem port)

Logical device:
`/dev/term/01`

Major device number:
128

Minor device number:
1

This channel has not got any restrictions and can be programmed like a normal tty.

The channel only considers one modem signal (M5, DCD).

`/dev/term/02`

Physical channel:
2681 – port B (modem port)

Logical device:
`/dev/term/02`

Major device number:
128

Minor device number:
2

This logical channel must be used for DIAL IN (remote login). Before it can be used, `ttymon` must be started on this device (login).

An opening to this device will block until the modem signals M1 (DSR) and M5 (DCD) are activated.

`/dev/term/03`

Physical channel:
2681 – port B (modem port)

Logical device:
`/dev/term/03`

Major device number:
128

Minor device number:
3

This logical device must be used for DIAL OUT (`cu`, `uucp`, etc).

An opening to this device will be successful without considering the modem signals M1 (DSR) and M5 (DCD). This device only accepts one opening at a time.

`/dev/term/04`

Physical channel:
2681 – port B (modem port)

Logical device:
`/dev/term/04`

Major device number:
128

Minor device number:
4

This logical device can be used for TELESERVICE or other services; the opening considers all modem signals set under the NVRAM address `0xbd000035` (byte access).

`0x01:`
M5 (DCD)

`0x02:`
M2 (CTS)

`0x04:`
M1 (DSR)

`0x10:`
M3 (CE)

16552 – PORT 0

Physical channel:
16C552 – Port 0

Logical device:
`/dev/term/05`

Major device number:
128

Minor device number:
5

This channel has not got any restrictions and can be programmed like a normal tty.

The control of the modem signals M1, M2, M5 and M3 can be set/reset under the NVRAM address `0xbd000038` (byte access).

`0x01:`
M5 (DCD)

`0x02:`
M2 (CTS)

`0x04:`
M1 (DSR)

`0x10:`
M3 (CE)

16552 – PORT 1

Physical channel:
16C552 – Port 1

Logical device:
`/dev/term/06`

Major device number:
128

Minor device number:
6

This channel has not got any restrictions and can be programmed like a normal tty.

The control of the modem signals M1, M2, M5 and M3 can be set/reset under the NVRAM address `0xbd000039` (byte access).

`0x01:`
M5 (DCD)

`0x02:`
M2 (CTS)

`0x04:`
M1 (DSR)

`0x10:`
M3 (CE)

IOCTL

The `sduart` driver supports the usual `ioctl`s for a tty. Additionally, the driver gets the following `ioctl`s with the form:

```
ioctl (fd, I_STR, &arg)
int fd
struct strioctl arg
```

The structure `strioctl` is defined in `<stropts.h>` and has to be filled in the following form:

```
ic_cmd    = ioctl request (TIOCMODS, TIOCMODG, TIOCMGET)
ic_timout = 0
ic_len    = sizeof(int modem_flags)
ic_dp     = (char *)&modem_flags
```

`modem_flags`

`0x00000001`: Modem signal M1 (DSR)
`0x00000002`: Modem signal M2 (CTS)
`0x00000004`: Modem signal M5 (DCD)
`0x00000008`: Modem signal M3 (CE)

`TIOCMODS`
Sets the modem signals to be controlled by the driver (M1, M2, M3, M5) into `modem_flags`.

`TIOCMODG`
Gets the currently modem signals to be controlled by the driver. The driver sets the signals (flags) into

`modem_flags`.

`TIOCMGET`
: The driver sets the current state of the modem signals into `modem_flags` and returns the state to the user.

FILES

`/dev/term/00` or `/dev/constty`
`/dev/term/01`
`/dev/term/02`
`/dev/term/03`
`/dev/term/04`
`/dev/term/05`
`/dev/term/06`
`/dev/console`
`/etc/ap/cons.ap`

SEE ALSO

`ldterm`(7), `termio`(7), `ttcompat`(7).
Programmer's Guide: STREAMS.

System Administrator's Reference Manual

sfdisk(7)
RM600 only

NAME

`sfdisk -` SCSI diskette (floppy disk)

DESCRIPTION

The SCSI diskette driver provides access to diskettes as both block and character devices. Diskettes must be formatted before their use [see `flformat`(1M)]. 3 ½" diskette format is supported. The driver controls up to two diskette drives. The minor device number specifies the drive number, the format of the diskette and the logical diskette number.

Diskette device file names (which correspond to a specific major and minor device) use the following format:

`/dev/ios0/[r]rsfdisk00{0-6}f{3e,3h,3d}[t]`

where `r` indicates a raw (character) interface to the diskette, `rsfdisk` selects the raw device interface and `sfdisk` selects the block device interface. The following list describes the possible formats:

`3e` 3½" extra-high-density diskette (2.88 mbyte).

`3h` 3½" high-density diskette (1.44 mbyte).

`3d` 3½" double-density diskette (720 kbyte).

If you can access the diskette to read and write, the format specification field can be omitted. In this instance, the diskette driver determines the existing format on the diskette and then executes the relevant operation (e.g. `cpio -itv </dev/ios0/rsfdisk005ft`).

The last parameter, `t`, selects the logical drive to be accessed. `t` represents the whole diskette. If `t` is not specified, the whole diskette except cylinder 0 will be selected.

The autodetection device automatically determines the format of the diskette in the drive, provided that the diskette corresponds to a standard format or it is unformatted. If an unformatted diskette is formatted with the autodetection device, it is formatted with the physical diskette format (displayed on the indicators – holes – of the diskette.

ioctl calls

The available `ioctl` calls and data structures are defined in `sys/sfd.h` and `sys/vtoc.h`.

ERROR MESSAGES

The driver retries failed transfers up to three times. If the request still has not succeeded, the driver displays an error message.

Attempts to write to a write-protected diskette are likewise reported.

If an I/O operation has to be repeated before it succeeds, the driver issues a message indicating that the drive or the diskette you are using is beginning to fail.

SEE ALSO

`flformat`(1M), `ioctl`(2), `fdsetup`(8).

System Administrator's Reference Manual

sgen(7)

NAME

sgen - SCSI generic device interface

DESCRIPTION

sgen is a generic interface that enables access to SCSI devices, which are not supported by other Reliant UNIX drivers. Like all other SCSI devices, generic devices are handled by the IOS subsystem.

SCSI devices are accessed via the special files in the directory /dev/ios0. SCSI generic devices are accessed via the raw (character) device only. Access via the block device will return ENXIO.

The naming convention for SCSI generic devices is as follows (all numbers are specified in decimal):

/dev/ios*b*/[r]sgen*ppu*

Where:

b Bus number (always 0).

r Designates the raw (character) device. If r is not specified, the device is a block device.

pp (RM600)
The logical controller number. The controller number is any 2-character decimal number between 00 and 99. The number indicates the device to which the SCSI bus is connected, and on which host adapter. Physical controllers are assigned to the logical controller numbers in /etc/controller.

pp (RM400)
The logical controller number (channel ID). The controller number is any 2-character decimal number between 00 and 99. The number indicates to which SCSI bus the device is connected to.

u The device's unit number (SCSI ID). The address may range from 0 to 15.

GENERIC DEVICES AS LOGICAL UNITS

If a SCSI device has several logical units (LUNs), the naming convention described above is valid for LUN 0. The following naming convention then applies for other LUNs:

/dev/ios*b*/rsgen*ppu*.*l*

l is a single-position decimal number and specifies the logical unit number of the device. All other name parts have the same meaning as described above.

Warning:

For reasons of backward compatibility, the addressing of logical units is only enabled (LUNs) via separate special files by default if the LUN 0 of the SCSI device is a tape device or if the device is connected to an FC/SCSI router (see further below). In other cases, there is only one special file – as in older Reliant UNIX versions – for the generic SCSI device, and the LUNs is addressed via 3 bits in the Command Descriptor Block of the sgen_escsicmd data structure [see sgenio(7)]. If LUNs addressing with separate special files is also required for such devices, the sgenmultilun constant must be set to 1 in the /etc/conf/pack.d/sgen/space.c file.

GENERIC DEVICES ON A ROUTER IN SCC MODE

The following naming convention applies for generic SCSI devices, which are connected with a router (e.g. Crossroads 4100) in SCC mode [see srout(7)] to a Fibre Channel Loop:

/dev/iof0/rsgen.*pp*.*nnnnn*.*ccu*.*l*

pp has the same meaning as described above.

nnnnn is a five-position character combination generated from the WWPN of the router (WWPN is a universally unique name for Fibre Channel devices).

cc is a two-position decimal number, which specifies the number of the SCSI bus on the router.

u is the number of the SCSI device on the SCSI bus of the router (value range 0 through 15, the values for 10 to 15 are represented hexadecimally as a through f).

l is a single-position decimal number, which specifies the logical unit number of the device.

GENERIC DEVICES ON A ROUTER IN AUTO ASSIGNED OR INDEXED MODE

If the router is not operating in SCC mode [see srout(7)], the following naming convention applies for LUN 0:

/dev/iof0/sgen.*pp*.*nnnnn*

The following naming convention then applies for all other LUNs:

/dev/iof0/sgen.*pp*.*nnnnn*.*l*

pp and *nnnnn* have the same meaning as above.

l is a single-position decimal number, which specifies the logical unit number used to identify the device in the router's address table.

ALTERNATIVE DEVICES

Alternatively, logical units can also be addressed with the aid of their entry in the device file system. Names in the device file system are structured as follows:

/devices/[r]*serialnumber*

The device numbers for logical units are assigned dynamically and are only known to the device file system [see devfs(7)].

In addition to the specified names, a symbolic name can be set up with the sgensetup(8) command for a generic SCSI device, assuming the device has a serial number.

MAJOR-MINOR DEVICE NUMBER

The major-minor device number can be determined using the /sbin/autoconf –d *device_name*.

The device (class) number is a 32-bit number, which has the following format for directly connected SCSI devices:

BBBMMMMM MMMMMMxx xxxxxxdd d-------

Where:

B Bus number: always 0.

M The major device number.

x The number of the host adapter and, if necessary, the numbers of the SCSI bus on the hardware [the number coding depends on the machine; for further information see sdisk(7)].

d The SCSI device address (0 to 15) on the specified SCSI bus.

Major-minor device numbers are assigned dynamically for SCSI devices on routers and for SCSI LUNs. In this case, the major-minor device numbers have no decodable format.

ERROR MESSAGES

In contrast to other SCSI device drivers, the generic SCSI device driver produces neither error messages nor log data. Error handling and logging must be performed by the application software.

CONFIGURATION

SCSI generic devices must be configured into the system by providing an entry in the device name table in /etc./conf/pack.d/sgen/space.c. Device name table entries have the following format (see <sys/sgenio.h>):

```
#define SGENDEVBYTES    16
#define SGENINQBYTES    24
#define SGENPNBYTES     64
struct  sgennames {
```

```
        char sgen_dname[SGENDEVBYTES]; /* Device node name */
        char sgen_iname[SGENINQBYTES]; /* Vendor and Product Id
                                          (max. 24 bytes) */
        char sgen_aname[SGENPNBYTES];  /* Pretty name for autoconf */
};
```

A device is configured as a generic SCSI device if the specifications in `sgen_dname` and in `sgen_iname` apply to the device. `sgen_dname` is the name of the device node as displayed by `autoconf(8)` (e.g. `ios0/sgen006`). When booting, the system compares the character string stored in the device table in `space.c` as `sgen_dname` to the character string that is displayed later by `autoconf(8)`. Only the number of bytes stored in `sgen_dname` are compared. The character "?" represents any character (e.g. `ios0/sgen??6`).

`sgen_iname` is the vendor and product identification from the SCSI inquiry command. The SCSI generic devices are identified by issuing the SCSI inquiry command. The vendor and product identification returned by the SCSI inquiry are matched against the inquiry names kept by the device name table. The name in `space.c` must match exactly, byte for byte, as otherwise, the device will not be identified as an "SCSI generic device". If fewer than 24 bytes are specified in `sgen_iname`, only those bytes are compared. Again the character "?" represents any character.

`sgen_aname` will be printed by `autoconf(8)` and other administration commands to identify the device.

SCSI generic devices are identified during booting or `iosreprobe(8)`. Devices which are configured for the SCSI generic device driver will be identified by the SCSI generic device driver only, even if there is an entry for this device in one of the device profile tables of the other SCSI device drivers.

An SCSI device with an SCSI device type that does not correspond to any SCSI device driver is configured for the SCSI generic device driver, even if there is no entry for this device in the device name table of the SCSI generic device driver.

It is recommended that you use the `sgenconf(8)` command for configuring generic SCSI devices.

The `autoconf(8)` command can be used to determine if a SCSI generic device is recognized by the system during booting.

Device nodes are created using the `/sbin/sgensetup` command. Refer to the `sgensetup(8)` manual page for more information.

FILES

`/dev/ios0/rsgen`*ppu*[`.`*l*]

`/dev/iof0/rsgen.`*pp*`.`*nnnnn*`.`*ccu*`.`*l*

`/devices/`[`r`]*serialnumber*

`/etc/conf/pack.d/sgen/space.c`

SEE ALSO

`devname(7)`, `sdisk(7)`, `sgenio(7)`, `sjuke(7)`, `stape(7)`, `sgenconf(8)`, `sgensetup(8)`.

System Administrator's Reference Manual

sgenio(7)

NAME

`sgenio` - SCSI generic device control interface

DESCRIPTION

This interfaces provides a way to send SCSI commands to SCSI devices, which are not supported by SNI. See `sgen(7)` for information about naming conventions and configuration of SCSI generic devices.

The raw (character) interface is the only interface, which is supported for SCSI generic devices. An attempt to open the block device will result in `ENXIO` error returned by the open routine. The only system calls supported are `open(2)`, `close(2)` and `ioctl(2)`. The system calls `read(2)` and `write(2)` are not possible on the SCSI generic device interface. An attempt to read or write will result in `ENODEV`.

Like all other SCSI devices, generic devices are probed and initialized at boot time or by the `iosreprobe(8)` command. During probe, up to 3 SCSI Test Unit Ready commands are sent to the device. If the device is present, a SCSI inquiry command is issued to the device.

At system call `open(2)` no actions are taken on the device. `O_EXCL` should be used to open the device exclusively. A request to open a device, which is already exclusively opened, or a request to exclusively open a device, which is already opened, will result in `EBUSY`.

If the device cannot be opened due to configuration problems, `ENXIO` will be returned.

At close no actions are taken on the device.

There is only one `ioctl` operation for SCSI generic devices. The `ioctl` operation has the following form:

```
#include <sys/sgenio.h>
struct escsicmd *p;
retval = ioctl(fildes, SGEN_ESCSICMD, p);
```

p is a pointer to a `escsicmd` structure:

```
struct  escsicmd {
      u_char  scsi0;          /* SCSI Byte 0 */
      u_char  scsi1;          /* SCSI Byte 1 */
      u_char  scsi2;          /* SCSI Byte 2 */
      u_char  scsi3;          /* SCSI Byte 3 */
      u_char  scsi4;          /* SCSI Byte 4 */
      u_char  scsi5;          /* SCSI Byte 5 */
      u_char  scsi6;          /* SCSI Byte 6 */
      u_char  scsi7;          /* SCSI Byte 7 */
      u_char  scsi8;          /* SCSI Byte 8 */
      u_char  scsi9;          /* SCSI Byte 9 */
      u_char  scsi10;         /* SCSI Byte 10 */
      u_char  scsi11;         /* SCSI Byte 11 */
      int     scs_flags;      /* Flags */
      char    *scs_addr;      /* Data Address */
      int     scs_bytes;      /* Maximum Number of Data Bytes */
      int     scs_timeout;    /* timeout in secs */
      int     scs_cdblen;     /* length of cdb for vendor unique cmds */
      short   scs_errtype;    /* Error Type */
      u_char  scs_scstat;     /* SCSI Status Code */
      u_char  scs_shaier;     /* Host Adaptor Error Code */
      int     scs_snslen;     /* max. length of sense data */
      char    *scs_snsaddr;   /* address of extended sense data buffer */
      int     dummy[5];       /* for future use */
};
```

`scs_errtype`, `scs_scstat` and `scs_shaier` are returned from the driver, all other parameters should be provided by the application software.

`scs_flags` is defined as follows:

```
#define SGENNODATA    0x01   /* No data for SCSI command */
#define SGENDATAIN    0x02   /* SCSI commands sends input data to device */
#define SGENDATAOUT   0x04   /* SCSI commands outputs data from device */
#define SGENNOTIMER   0x08   /* Disable software timeout timer */
#define SGENQUIET     0x20   /* Suppress error messages from HA driver */
#define SGENAUTORQS   0x200  /* Auto Request Sense (ext. scsi cmds only) */
#define SGENCAMCODES  0x400  /* return CAM error codes (see ios_conf.h) */
```

If `SGENDATAIN` or `SGENDATAOUT` are set, the data address and the maximum number of data bytes have to be specified in `scs_addr` and `scs_bytes` respectively.

`scs_timeout` specifies the timeout interval for this SCSI operation in seconds. It is recommended not to use timeout intervals smaller than 10 seconds.

`scs_cdblen` specifies the length of the command descriptor block. If `scs_cdblen` is zero the driver sets the command descriptor block length according to SCSI specifications or according to driver internal rules in case of vendor unique commands.

`scs_errtype` specifies, which type of error has occurred. This parameter has to be checked, if `EIO` is returned. Following definitions are from `<sys/sgenio.h>`:

```
#define SGENER_GOOD           0       /* No Error */
#define SGENER_TIMEOUT        1       /* Timeout */
#define SGENER_CCERR          2       /* Check Condition */
#define SGENER_ERROR          3       /* Other errors */
```

`scs_scstat` is the SCSI status returned to the host adapter. Following definitions are from `<sys/ios_cmd.h>`:

```
#define SCSTAT_GOOD           0x00    /* Good Status */
#define SCSTAT_CHECKCOND      0x02    /* Check Condition */
#define SCSTAT_BUSY           0x08    /* Busy Status */
#define SCSTAT_IGOOD          0x10    /* Intermediate/Good Status */
#define SCSTAT_RCONFLICT      0x18    /* Reservation Conflict */
```

`scs_shaier` is the host adapter error code. There are a lot of host adapter error codes (see `<sys/ios_conf.h>`). Note that most of them are host adapter specific and the meaning may differ for different host adapters. Following definitions from `<sys/ios_conf.h>` are used by all host adapters:

```
#define SHAIER_GOOD         0x00    /* Good status, no error */
#define SHAIER_CMDUA        0x09    /* Command Unavailable */
#define SHAIER_PARITY       0x0a    /* Parity Error */
#define SHAIER_RSTBUS       0x11    /* Reset Bus Status */
#define SHAIER_CMDABRT      0x15    /* Command Aborted by Reset */
#define SHAIER_VMETMO       0x21    /* Memory Timeout Error */
#define SHAIER_FCHERR       0x26    /* Memory Fetch Error */
#define SHAIER_SCSTMO       0x30    /* SCSI Selection Timeout Error */
#define SHAIER_SCSI         0x32    /* SCSI Error */
#define SHAIER_SCBSEQ       0x33    /* Illegal SCSI Bus Sequence */
#define SHAIER_SCCNT        0x34    /* SCSI Transfer Count Exception */
```

The SCSI status "Check Condition" is usually interpreted by the driver as an error and is reported as `EIO`. If you set `SGENAUTORQS` in `scs_flags`, the SCSI command "Request Sense" is automatically issued by the driver when a "Check Condition" occurs. In this case, the parameters `scs_snsaddr` and `scs_snslen` must also be set. `scs_snsaddr` is the address of a data buffer where the driver can copy the sense data, `scs_snslen` specifies the length of this data buffer in bytes.

If the SCSI command has been executed successfully, `ioctl` returns 0 and `errno` is not set. If the command has not been executed successfully, `ioctl` returns –1 and `errno` is set to an appropriate value (see **DIAGNOSTICS**).

DIAGNOSTICS

Following error codes are generated by the device driver `ioctl` interface:

EINVAL
: `ioctl` code invalid.

EFAULT
: Illegal user address.

ENXIO
: Incorrect configuration or input parameters out of range.

EIO
: Device error, host adapter error or timeout.

NOTES

In addition to the `ioctl` operation `SGEN_ESCSICMD` described here, the `ioctl` operation `SGEN_SCSICMD` will still be provided for reasons of downwards compatibility. The SCSI command "Request Sense" can only be issued automatically by the driver (`SGENAUTORQS`) with the `ioctl` operation `SGEN_ESCSICMD`.

FILES

`/dev/ios0/rsgen`*ppu*

SEE ALSO

`devname`(7), `sgen`(7), `autoconf`(8), `sgensetup`(8).

System Administrator's Reference Manual

si(7)
RM600 only

NAME

`si` - serial interface multiplexer for MBII and EISA

DESCRIPTION

The `si` driver supports up to 16 SIM controllers (SIM = Serial Interface Multiplexer). Each SIM controller drives 6 AFP two-wire lines (AFP = alternating edge pulse transmission). Each of these lines can have up to 8 terminal attachment concentrators (TAKs). Up to eight devices may be connected to one TAK. The driver can support a maximum of 64 devices per controller. The minor device numbers 0–63 address SIM0, 64–127 address SIM1, 128–191 address SIM2, and so on.

In addition to the `si` driver, an SIM administration driver `sv` is provided, which is used to download the SIM and TAK controllers, and to provide diagnostic information. The device names are:

device name	minor	usage
`/dev/sim/smsload`*board#*	64**board#*+0	SIM load device
`/dev/sim/smsdiag`*board#*	64**board#*+1	SIM diagnostics device
`/dev/sim/smtload`*board#*	64**board#*+2	TAK load device
`/dev/sim/smtdiag`*board#*	64**board#*+3	TAK diagnostics device
`/dev/sim/smadmin`*board#*	64**board#*+4	administration device

When booting, software is downloaded to the SIM controller by the utility `simload`. The name of the downloaded program is `simsw`. A file `SIMCONFIG` is used to hold all the configuration information. This file is normally created using the administration menu system. The `Tak` utility is used to ascertain the firmware versions of the TAK, download the TAK, dump send/receive buffers, dump the TAK-RAM, determine the number of HDLC (HDLC = High Level Data Link Control) retries, change the HDLC window/frame length, etc. The `simmsg` utility is used to configure and deconfigure devices on the SIM/TAK controllers. All the TAK and SIM utilities are found in the `/sbin/sim` and `/etc/sim` directories.

The `SIMCONFIG` file for a machine, where boards 2 and 8 are plugged in, can thus look as follows:

SI2 at /dev/sim

MINORDEV	SIM	LINE	TAK	TAKCHANNEL	TAKNAME
128	2	0	1	0	tak1
129	2	0	3	0	tak3
130	2	1	4	0	tak4
131	2	0	1	1	tak1
132	2	0	1	2	tak1
133	2	0	1	4	tak1
134	2	0	3	1	tak3
135	2	0	3	4	tak3
136	2	0	3	5	tak3
137	2	1	4	1	tak4
138	2	1	4	4	tak4
139	2	1	4	5	tak4

SI8 at /dev/sim

MINORDEV	SIM	LINE	TAK	TAKCHANNEL	TAKNAME
512	8	0	1	1	CCtr1
513	8	0	1	2	CCtr1
514	8	0	1	0	CCtr1
515	8	1	2	0	CCtr4
516	8	1	2	1	CCtr4
517	8	1	2	4	CCtr4
518	8	1	2	5	CCtr4
519	8	0	3	0	Room231
520	8	0	3	1	Room231
521	8	0	3	4	Room231
522	8	0	3	5	Room231
523	8	1	4	0	Room232
524	8	1	4	1	Room232
525	8	1	4	4	Room232
526	8	1	4	5	Room232

The entries after TAKCHANNEL are not relevant for the driver. They are used to configure the TAKs. For reasons of compatibility, columns should not be inserted manually.

The program `simload` searches the file `SIMCONFIG`; for each entry `SI`*board#* at `/dev/sim`, it loads the SIM controller via the load device `/dev/sim/smadmin`*board#*, starts the on-board software, and sends a configuration message to the controller for each minor device number.

During startup, `simload` is called by the `/etc/rc` shell script without any parameters. This loads all controllers and configures all devices.

The administration menu system calls `simload` as follows:

`simload` [`–s`] *board#*

This loads only controller `board#`. The `–s` option (synchronous) sends the configuration messages one after another, and not in parallel.

The configuration of a SIM controller can be changed without influencing other devices connected to the same SIM/TAK using the program `simmsg`:

`/sbin/sim/simmsg "config` *path minor# line# tak# chan#*`"`

Calling this program with *line#* = *tak#* = *chan#* = 255 deconfigures a device. Calling the program again with valid numbers for *line#*, *tak#*, and *chan#* configures a new device. Reconfiguration is carried out by first deconfiguring a device and then configuring it again.

Valid numbers for *line#* are 0 to 5.

Valid numbers for *tak#* are 1 to 9 (0 is reserved for hardware tests).

For old TAKs with SS97 interfaces valid numbers for *chan#* are 0 to 3 for S0 to S3, and 4 to 7 for V0 to V3 without modem functionality. For new TAKs which only have V24 interfaces, valid numbers for *chan#* are 0 to 7 for V0 to V7 without modem functionality. If modem functionality is to be supported for a channel, then the bit `0x80` must be set in *chan#*.

C programs can access this utility via the system(3S) library call.

Example

```
#define DEVS_PER_SIM    16
char conf_path[17] = "/dev/sim/smadmin";
/* -- sim_config: configure terminal with minor 'minor_dev' */
int
sim_config(
 int minor_dev,
 unsigned char line_nr,
 unsigned char tak_nr,
 unsigned char chn_nr)
{
 char buf[80];

 sprintf(buf, "/sbin/sim/simmsg 'config %s%d %d %d %d %d'",
  conf_path,
  minor_dev / DEVS_PER_SIM,
  minor_dev,
  line_nr,
  tak_nr,
  chn_nr);
 return(system(buf)8);
}
```

ioctl Calls

All the standard `ioctl` calls for terminals drivers (`TCSETA`, `TCGETA`, etc.) work via the XPGIII/POSIX library functions `tcsetattr()`, `tcgetattr()`, etc. The following call is also provided:

```
TIOCRTSCTSFLOW
ioctl(f, I_STR, (char *)&sti)
struct strioctl sti;
sti.ic_cmd = TIOCRTSCTSFLOW;
sti.ic_timeout = 0;
sti.ic_len = sizeof(char);
sti.ic_dp = &arg;
char arg;
```

This call is used by the spooler subsystem to control line printers which do not use the usual `XON/XOFF` protocol, but which control data flow via the RTS/CTS (RTS = Ready To Send; CTS = Clear To Send) signal lines. The `ioctl` argument is the address of a `strioctl` structure as described in `stropts.h`. A value of 1 for `arg` sends a message to the TAK to switch on S2/M2 flow control between TAK and SIM. A value of 0 sends a message to the TAK to switch off S2/M2 flow control. This I/O control is not intended as an interface for an end user, its usage is restricted to the SNI spooler subsystem, and it may be modified or replaced by another interface in the future.

DIAGNOSTICS

The SIM controller returns an error status byte in each message. This status consists of 2 bytes: The lower byte contains the actual error information, and the upper byte contains an error code from the HDLC layer, if available.

The following error codes are defined:

`MST_OK 0x00`
: No error.

`MST_INVAL 0x01`

Invalid command. An undefined command code was found in a message from the driver.

`MST_ERR 0x02`
: Command cannot be executed. The reason for this error code depends on the command. It indicates, for example that a parameter is out of range in a config message, or that the amount of data in a `MUX_SEND` does not match the byte count indicated in the message. This error is also returned if the driver tries to open a device on the SIM that is already open, or issues a command for a device that is not open, or in an inappropriate state.

`MST_NODVC 0x03`
: Device not configured. The `CONFIGURE_DEVICE` message is seen on the SIM board for the respective device. This may happen if an application tries to open a device where the SIM controller has just been reloaded but where the simload script has not yet sent the config message to the board.

`MST_TAKINOP 0x04`
: TAK cannot be reached. This status indicates that the HDLC connection to the TAK is not yet established (e.g. because the TAK is not on or the data line is interrupted). In this case, the upper byte is 0.

  This status is also returned if the HDLC connection has crashed. In this case, the upper byte contains a reason code from the HDLC layer to indicate why the connection went down (see below).

`MST_DEAKT 0x05`
: SIM is being deactivated. This status is returned if commands are not executed because of a pending deactivation of the SIM controller, usually caused by a `simload` command.

`MST_LINEINOP 0x06`
: Line is inoperable. A severe error occurred in the initialization of a HDLC connection. All devices that were configured for this line are deconfigured and are unavailable.

`MST_CANCEL 0x07`
: Command canceled by SIM. This status is used to terminate commands that are still pending when a device is closed. E.g. `INPUT_GO` or `MUX_RECEIVE` commands are terminated in this manner, and the driver should not try to reissue these commands.

The following error codes may be returned by `MST_TAKINOP` above:

`RC_TERM 0x10`
: Connection terminated while trying to connect

`RC_PINOP2 0x20`
: Connection lost, retry counter expired

`RC_SOP2 0x30`
: SNRM/DM received (secondary lines only)

`RC_PDISC2 0x40`
: Connection terminated upon request

`RC_DMIN2 0x50`
: DM Frame received from secondary TAK

`RC_FRMRI2 0x60`
: FRMR Frame received from secondary TAK

`RC_NEFIN2 0x80`
: Invalid Frame received from secondary TAK

`RC_DISCI2 0x90`
: DISC received

`RC_RDPI2 0xB0`
: Retry counter expired after RD was received

`RC_NOPOLL 0xC0`
: Primary did not poll (secondary lines only)

`RC_RDIN2 0xF0`
: RD frame received from secondary

Recovery Actions

The lower half-byte of the error code is most important for resolving the problem

`<ERROR>=0x1`

This error code probably means that the SIM driver (Reliant UNIX driver) has a bug. If this happens, contact the driver developer. This error code was not previously displayed.

`<ERROR>=0x2`

The SIM driver and software are not compatible anymore, whether the corresponding tty-interface is opened or not. You should contact the SIM driver developer and the SIM software developer. You can temporarily resolve the problem by reloading the relevant SIM board. This can be done using the command

`/sbin/sim/simload` *logical_board-number*

`<ERROR>=0x3`

A device node exists, but there is no corresponding entry of the following kind in the file `/etc/sim/SIMCONFIG` of the following kind:

```
#MINDEV SIM LINE TAK TAKCHANNEL
067      2   1    5    7
```

This may be due to a bug in the administration system. To fix the problem temporarily, remove the device node and try to configure the tty interface again with the help of the administration system.

`<ERROR>=0x4`

If this error code is displayed, check the status of the TAK. In the simplest case, the TAK is not turned on. If the TAK is on, check that the orange light on the TAK is on. This means that the TAK is being polled by the SIM software. Next, check that no error occurred during configuration. Check that the TAK is connected to the line contained in the `/etc/sim/SIMCONFIG` file, and that the HDLC address is the same as the one contained in that file. If this is so, there are two reasons for the error message:

1. The AFP line is damaged. Check the AFP line to the TAK.
2. The TAK is damaged. Replace the TAK.

If the problem does not originate with either the status of the TAK, or the AFP line, take a dump of the SIM board using the following command:

`mb2dump -b /dev/sim/smsload`*CTLR_NO* `0 0xfffff \`
`DUMP_`*CTLR_NO*

and send the `DUMP_`*CTLR_NO*-file to the SIM software developer. This dump contains statistical information on problems relating to the HDLC line which may help fix the problem. It is best to have an AFP tracer which can check the status of the AFP line and report problems immediately.

`<ERROR>=0x5`

This error code has never been displayed. This error code should not be displayed for the following reasons:

- The SIM board will be deactivated by the SIM software only in the following circumstances:
  - The most recently configured interface on the SIM board is deconfigured (with permission from the administration system).
  - The SIM board becomes overloaded and SIM driver sends a disable message to the SIM software on the relevant SIM board.
- If the SIM software deactivates the SIM board, it sends a disable message up to the SIM driver, either unrequested, or in response to a disable message sent by the SIM driver.
- The SIM driver does not send any command other than configuration commands after it has received the disable message.

Reloading the relevant SIM board with

`/sbin/sim/simload`

should rectify the problem.

However, contact the developer of the SIM driver and software for information on other diagnostic procedures.

`<ERROR>=0x6`

This error code was not previously displayed. Take a dump of the relevant SIM board using the command

`mb2dump -b /dev/sim/smsload`*CTLR_NO* `0 0xfffff \`

`DUMP_`*CTLR_No*

and send the `DUMP_`*CTLR_NO*-file to the SIM software developer. To temporarily rectify the problem, load the relevant SIM board with

`/sbin/sim/simload` *CTLR_NO*

`<ERROR>=0x7`

This error code can occur if a process has set up an `ioctl`, is waiting for a response to a `param` or `drain` and is then signaled. Normally, no action is required. However, it could be that the configuration time that the ttymon waits for `ioctl`'s to to be configured is too short, and so the `ttymon` sends a signal to the process. If this happens, reconfigure the `ttymon`.

NOTES

If the actions outlined above do not help to rectify the problem, use the AFP tracer to find the problem in the AFP connection.

FILES

`/dev/term/tty*`

`/sbin/sim/simload`

`/etc/sim/SIMCONFIG`

SEE ALSO

`ttymon`(1M), `signal`(2), `termios`(3C), `termio`(7).

System Administrator's Reference Manual

sjuke(7)

NAME

`sjuke` - SCSI jukebox interface

SYNOPSIS

`/dev/ios` *b*`/[` `r` `]sjuke`*ppu*

DESCRIPTION

`sjuke` provides access to SCSI jukebox devices. SCSI jukebox devices are accessed through the special files in the `/dev/ios0` directory. SCSI jukeboxes are accessed through the raw (character) device only. Access through the block tape device will return `ENXIO`.

The naming convention for SCSI jukebox devices is as follows (all numbers are specified in decimal form):

`/dev/ios`*b*`/[r]sjuke`*ppu*

b Bus number (always 0).

`r` Designates the raw (character) device.

pp (RM600)
The logical controller number. The controller number is any 2-character decimal number between 00 and 99. The number indicates which SCSI bus the device is connected to on which host adapter. Physical controllers are assigned to the logical controller numbers in `/etc/controller`.

pp (RM400)
The logical controller number (channel ID). The controller number is any 2-character decimal number between 00 and 99. The number indicates to which SCSI bus the device is connected.

u The device's unit number (SCSI ID). The address may range from 0 to 15.

MAJOR-MINOR DEVICE NUMBER

The major-minor device number can be determined via `/sbin/autoconf -d` *device_name*.

The (32-bit) major-minor device number is defined as follows:

BBBMMMMM MMMMMMxx xxxxxxdd duuu`----`

B Bus number: always 0.

M The major device number.

x The number of the host adapter and if necessary the numbers of the SCSI bus on the hardware [the number coding depends on the machine, for more information see `sdisk(7)`].

d The SCSI device address (0 to 15) on the specified SCSI bus.

u The SCSI logical unit number of the specified SCSI device, if supported. Most devices do not use LUNs, therefore bits 4 – 6 will usually be 0.

ERROR MESSAGES

The error messages displayed for the SCSI jukebox devices have the following format:

```
ios0/sjuke002: <status> 053b0200, <cstat> a5053b0e Illegal Request - Source empty
```

where:

`ios0/sjuke002:`
The SCSI jukebox device that had an error. In this case, the jukebox device located at device address 2.

`<status>` *KKAASSCC*
The SCSI composite status code. The composite status code has the following format:

KK The SCSI Sense Key value.

AA The SCSI Additional Sense Key value.

SS The SCSI status returned to the SCSI host adapter.

CC The SCSI host adapter error code.

`<cstat>` *OOKKAAQQ*

The SCSI additional status code. The additional status code has the following format:

OO SCSI operation code.

KK SCSI sense key value.

AA SCSI additional sense key value.

QQ SCSI additional sense qualifier.

`Illegal Request`

The decoded SCSI sense key message.

`Source Empty`

The decoded SCSI additional sense key message.

ERROR LOGGING

Generally, device errors are logged in a file named

`/var/adm/error/errfile.current.Z`.

Older error files are named `errfile.previous.Z` and then `errfile.`*date* (for example: `errfile.03.05-15:17.Z`). Error reports for specific device types can be viewed by issuing appropriate filters. For example, error reports for jukebox devices can be viewed by issuing the command:

```
zcat /var/adm/error/errfile.current.Z | /var/adm/error/bin/pyr_iosjuke
```

The command `pyr_iosjuke` searches for jukebox error reports and prints them in readable format.

SCSI JUKEBOX CONFIGURATION

Device nodes are created using the `/sbin/jbsetup` command. Refer to the `jbsetup(8)` manual page for more information.

SCSI jukebox devices are identified during booting. The jukebox type is determined by issuing the SCSI inquiry command. Bytes 8 to 31, the vendor and product identification, returned by the SCSI inquiry are matched against the inquiry names kept by the SCSI jukebox profile tables defined in `/etc/conf/pack.d/acc/space.c`. The name in `space.c` must match exactly, byte for byte, otherwise, the jukebox will be classified as "unknown."

The `autoconf(8)` command can be used to determine if a jukebox device is recognized by the system when it is being rebooted. For example, using `autoconf -l`:

```
ios0/sjuke014 9083-20 (Jukebox)
```

appears for a 9083-20 jukebox recognized by the system.

```
ios0/sjuke014: not configured - Unknown Jukebox
```

appears for a jukebox device the system did not recognize. This device would be classified as "unknown" by the driver.

An "unknown" jukebox device is any device that does not have a jukebox profile defined. Default parameters are used to allow the device to function. However, some jukebox devices require specific bits to be set in order to function correctly. If those jukebox devices are not recognized and are classified as "Generic", they may not function properly. It is suggested that all jukebox devices are configured onto the system using the appropriate tables, so the system recognizes the device correctly at boot.

Drive specific parameters are set during jukebox open with a SCSI "mode select" command. These parameters are specified by the default "mode select" parameter entry for the jukebox type in `/etc/conf/pack.d/acc/space.c`.

FILES

`/dev/ios`*b*`/[r]sjuke`*ppu*
`/etc/conf/pack.d/acc/space.c`

SEE ALSO

`accioctl`(7), `devname`(7), `stape`(7), `jbsetup`(8).

System Administrator's Reference Manual

sm_cons(7)
RM600 only

NAME

`sm_cons` - interface to special commands of the CSImaintenance task

DESCRIPTION

The device entry `/dev/sm_cons` offers an interface to specific regions of the CSI NVRAM. These regions comprise the boot parameters that can be read and changed by an `ioctl` call, and the console ring buffer the contents of which are returned by reading `/dev/sm_cons`.

It is also possible to switch between the two operating modes of the 25-pin modem port of the CSI board: Tele-X-Link and remote console. The current operating mode can be determined and whether or not a connection is established. The dial string for remote console can be set or displayed. The parameter for terminal setting of the teleservice port can be set or displayed in/from CSI NVRAM. The current or saved values of the console ring buffer stored in the CSI NVRAM can be read.

The following `ioctl` calls exist:

```
#include <sys/io/sm_ioctl.h>
#include <sys/termio.h>
int ioctl(int fildes, SMIOSTATS, struct sm_stats *arg);
int ioctl(int fildes, int request, struct ioctl_reboot *arg);
int ioctl(int fildes, int request, struct sm_modem *arg);
int ioctl(int fildes, int request, struct termio *arg);
int ioctl(int fildes, int request, char *arg);
```

Subsequently a description of the specific calls is given.

sm_cons call statistics

```
int ioctl(int fildes, SMIOSTATS, struct sm_stats *arg);
```

The parameters are:

fildes
: File descriptor returned when the file was opened.

request
: Device-dependent function code `SMIOSTATS`.

arg Pointer to a device-dependent data structure of the following type:

```
struct sm_stats
{
    int     sm_stats_cmd;       /* number of commands */
    int     sm_stats_ioctls;    /* number of ioctls */
    int     sm_stats_xfers;     /* number of transfers */
}
```

The structure `sm_stats` contains the number of commands, `ioctl` and read calls since the last system boot.

Read and set boot parameters

```
int ioctl(int fildes, int request, struct ioctl_reboot *arg);
```

The parameters are:

fildes
: File descriptor returned when the file was opened.

request

One of the following device-dependent function codes:

`SMIOGETREBOOT`
: to read boot parameters

`SMIOSETREBOOT`
: to set boot parameters

arg Pointer to a device-dependent data structure of the following type:

```
struct ioctl_reboot
{
   int re_powerup;                /* 1: permanent values
                                   * 0: temporary values  */
   struct boot_data re_bd;        /* layout of boot flags,
                                   * boot and dump string */
}
```

The structure struct `boot_data` is built up as follows:

```
struct boot_data {
       uint_t      bd_boot_flags;
       char        bd_boot_string[80];
       char        bd_dump_string[80];
}
```

For a detailed description of boot flags and boot strings, please refer to `bootflags`(8). `bd_dump_string` is currently not used.

Modem operating mode commands

```
int ioctl(int fildes, int request, struct sm_modem *arg);
```

The parameters are:

fildes
: File descriptor returned when the file was opened.

request
: One of the following values is allowed:

`SM_SWITCHTXTORC`
: Switch modem operating mode from Tele-X-Link to remote console mode.

`SM_SWITCHRCTOTX`
: Switch modem operating mode from remote console mode to Tele-X-Link mode.

`SM_GETMODEMSTAT`
: Get current operating mode and connection state.

`SM_SETPERMBITS`
: Set/clear permission mode for remote console and/or dial in.

The structure referred to by the third parameter is filled with the current operating mode and connection state after the completion of any of the four `ioctl` calls above.

arg Pointer to a device and request-dependent data structure of the following type:

```
struct sm_modem {
 char modem_mode;      /* modem operating mode  */
 char modem_state;     /* connection state      */
 char remco_perm;      /* Perm.: remote console */
 char dialin_perm;     /* Perm.: dial in        */
}
```

This structure contains the current modem operating mode and connection state. For the first two calls, this is the state resulting from the switch. The two possible operating mode values are:

`SM_TX`

Tele-X-Link mode is on

`SM_RC`
remote console mode is on

The connection state is coded as follows:

`SM_ACTIVE`
amodem connection is established

`SM_NOTACTIVE`
no modem connection at present

The permission values are coded as follows:

`PERM_OFF`
permission denied

`PERM_ON`
permission allowed

The `SM_SETPERMBITS` request sets the permission mode to the specified value of `remco_perm`/`dialin_perm`. If the value differs from `PERM_OFF`/`PERM_ON` no modification is done.

Get/set dial string of remote console

```
int ioctl(int fildes, int request, char *arg);
```

The parameters are:

fildes
File descriptor returned when the file was opened.

request
One of the following values is allowed:

`SM_GETDIAL`
get dial string of remote console

`SM_SETDIAL`
set dial string of remote console

The structure referred to as the third parameter is filled with the current dial string of remote console.

arg Pointer to null-terminated character string. Maximum number of characters is `MAX_DIAL_CHARS` (defined in `<sys/io/sm_ioctl.h>`). The character "?" can be used as a placeholder. Maximum number of placeholders is 5.

Get/set terminal parameter for teleservice port

```
int ioctl(int fildes, int request, struct termio *arg);
```

The parameters are:

fildes
File descriptor returned when the file was opened.

request
One of the following values is allowed:

`TCGETA`
get terminal parameter

`TCSETA`
set terminal parameter

`TCSETAW`
set terminal parameter

The structure referred to as the third parameter is filled with the current terminal setting parameters for the teleservice port.

arg Structure contains terminal parameter for teleservice port [refer to `termio(7)`]. The following terminal

parameters may be set:

- character frame
- parity type
- parity mode
- stopbits
- baudrate
- xon/xoff protocol

Other parameter settings are without any effect. For detailed information of the `termio` structure refer to `termio(7)`.

Get content of console ring buffer

```
int ioctl(int fildes, int request, char *arg);
```

The parameters are:

fildes
: File descriptor returned when the file was opened.

request
: One of the following values are allowed:

`SM_READSAVNVRAM`
: read content of console ring buffer saved from last reboot

`SM_READCURNVRAM`
: read current content of console ring buffer

The `SM_READSAVNVRAM` request works only once after reboot, after which the buffer saved is then no longer available and an error is returned.

arg Pointer to character array. This character array (size = `CRB_NVRAM_SIZE`, defined in `<sys/io/sm_ioctl.h>`) must be allocated by the requesting user.

DIAGNOSTICS

`ioctl` returns NULL if successful, −1 in case of an error; *errno* contains the corresponding error code.

The error codes are:

`EBADF`
: *fildes* is not a valid open file descriptor.

`EINVAL`
: *request* or *arg* is not valid.

`EPERM`
: The effective user ID is not superuser.

`EFAULT`
: An error occurred when copying data from user to kernel address space or vice versa.

`EIO` Error in reading or writing data from and to the NVRAM.

`ENXIO`
: An invalid function code was delivered.

FILES

`/dev/sm_cons`

SEE ALSO

`csimaint(7)`, `bootflags(8)`.

System Administrator's Reference Manual

srout(7)

NAME

`srout` - interface for fibre channel-to-SCSI router

DESCRIPTION

`srout` is an interface that enables access to fibre channel-to-SCSI router (e.g. Crossroads 4100). With the aid of a router, SCSI devices can be connected to an FC (Fibre Channel) loop. Generally, no requests are sent to the router itself, rather to the connected SCSI devices [see the notes on logical units in `sgen(7)` and `stape(7)`].

The router itself can be accessed in SCC mode (SCSI-3 Controller Commands). One of the following two devices is used for this purpose:

`/dev/iof0/rsrout.`*pp*`.`*nnnnn*

or

`/devices/r`*serialnumber*

where

pp is the logical controller number. This is a two-position decimal number in the range 00 through 99. The number specifies the FC loop to which the device is connected.

nnnnn is a 5-position character combination generated from the WWPN of the router (the WWPN is a universally unique name for FC devices).

DEVICE(CLASS)NUMBER

The device number for routers is assigned dynamically and is only known to the device file system [see `devfs(7)`].

CONFIGURATION

The `autoconf(8)` command can be used to establish whether a router is detected by the system when it boots. The device nodes are created with the `/sbin/sroutsetup` command. For further details, please refer to `sroutsetup(8)` .

NOTES ON ROUTERS OF TYPE CROSSROADS 4100

If an additional SCSI device is to be connected to a Crossroads router,

1) the new device must be connected and enabled,
2) the router must be switched off and then directly on again,
3) the `changedev –D` command must be invoked,
4) the new device must be made known to the system with the `reinitdev(8)` command.

Crossroads routers can be configured using a serial interface or via `telnet`. Among other things, this interface offers a choice of different addressing types:

- SCC Addressing
- Indexed Addressing
- Auto Assigned Addressing

To change the type of addressing,

1) invoke the `changedev –D` command for the router
2) reconfigure the router (serial interface or `telnet`)
3) invoke the `reinitdev` command

It is not necessary to reconfigure the operating system kernel in order to change the type of addressing.

NOTES ON AUTO ASSIGNED AND INDEXED ADDRESSING

These operating modes must be selected in heterogeneous multihost configurations if remote hosts do not support SCC addressing. The following peculiarities must be noted for these addressing types:

- The router is not visible in the `autoconf` tree and cannot be addressed as a device
- A reset to the device with the LUN number 0 causes a reset on all other devices on the same router
- The SCSI ID of the device cannot be derived from the `autoconf` name
- The LUN numbers of the devices may change in Auto Assigned mode with a reboot

FILES

`/dev/iof0/rsrout.`*pp.nnnnn*

`/devices/`*rserialnumber*

`/etc/conf/pack.d/srout/space.c`

SEE ALSO

`sgen`(7), `stape`(7), `sroutsetup`(8).

System Administrator's Reference Manual

stape(7)

NAME

stape - SCSI tape interface

DESCRIPTION

stape provides access to SCSI tape devices. SCSI tape devices are accessed through the special files in the /dev/ios0 directory. SCSI tape drives are accessed through the raw (character) device only. Access through the block tape device will return EINVAL.

The naming convention for SCSI tape devices is as follows (all numbers are specified in decimal):

/dev/ios*b*/[r]stape*ppu*[c][h][n][o][v][g]

b Bus number (always 0).

r Designates the raw (character) device. If r is not specified, the device is a block device.

pp (RM600)
The logical controller number. The controller number is any 2-character decimal number between 00 and 99. The number indicates which SCSI bus the device is connected to on which host adapter. Physical controllers are assigned to the logical controller numbers in /etc/controller.

pp (RM400)
The logical controller number. The controller number is any 2-character decimal number between 00 and 99. The number indicates which SCSI bus the device is connected to on which host adapter.

u The SCSI tape drive number. This number ranges from 0 to 15. For values 10 to 15 the hex representation a to f will be used. It is important that the device address does not conflict with the SCSI ID of the host adapter.

c If specified, designates the compression density device.

h If specified, designates the high density device.

n Designates the no-rewind on close device.

o Designates a device that does not rewind on closing and also does not move forward to the next tape mark.

v Designates the variable block mode for devices which allow both variable *and* fixed block mode.

g For these devices, the generic ioctl interface for reading and writing by means of ioctl(2) system calls can be used in the tape driver.

Note:

A higher level of recording density or data compression is not supported by all devices. If data is output via the directory entry of one of these devices, the capacity of the tapes is not increased. The generic interface for reading and writing to the drive by means of ioctl(2) system calls, with which the corresponding ioctl calls are forwarded directly to the device, is only supported for certain high-performance drives.

TAPE DEVICES AS LOGICAL UNITS

If a SCSI device has several logical units (LUNs), the naming convention described above is valid for LUN 0. The following naming convention then applies for other LUNs:

/dev/ios*b*/[r]stape*ppu*.*l*[c][h][n][o][v][g]

l is a single-digit decimal number, which specifies the logical unit number of the device. All other parts of the name have the same meaning as described above.

TAPE DEVICES ON A ROUTER IN SCC MODE

The following naming convention applies for SCSI tape devices, which are connected with a router (e.g. Crossroads 4100) in SCC mode [see srout(7)] to a Fibre Channel Loop:

/dev/iof0/[r]stape.*pp*.*nnnnn*.*ccu*.*l*[c][h][n][o][v][g]

pp and the device options (c, h, n etc.) have the same meaning as described above.

nnnnn is a five-digit character string generated from the WWPN of the router (the WWPN is a universally unique name for fibre channel devices).

cc is a two-digit decimal number, which specifies the number of the SCSI bus on the router.

u is the number of the SCSI tape drive on the SCSI bus of the router (value range: 0 to 15, the values for 10 to 15 are represented in hexadecimal notation as a through f).

l is a single-digit decimal number, which specifies the logical unit number of the device.

TAPE DEVICES ON A ROUTER IN AUTO ASSIGNED OR INDEXED MODE

If the router is not operating in SCC mode [see srout(7)], the following naming convention applies for LUN 0:

/dev/iof0/sgen.*pp*.*nnnnn*

The following naming convention then applies for all other LUNs:

/dev/iof0/sgen.*pp*.*nnnnn*.*l*

pp and *nnnnn* have the same meaning as above.

l is a single-position decimal number, which specifies the logical unit number used to identify the device in the router's address table.

ALTERNATIVE DEVICES

Alternatively, logical units can also be addressed be means of their entry in the device file system. Names in the device file system are structured as follows:

/devices/[r]*serialnumber*.[c][h][n][o][v][g]

The device numbers for logical units are assigned dynamically and are only known to the device file system [see devfs(7)].

In addition to the names mentioned, a symbolic name can also be set up for a tape device using the tpsetup(8) command if the device has a serial number.

MAJOR-MINOR DEVICE NUMBER

The major-minor device number may be determined via the /sbin/autoconf –d *device_name*.

The device (class) number is a 32-bit number, which has the following format for directly connected SCSI devices:

BBBMMMMM MMMMMMxx xxxxxxdd duuvgCnh

B Bus number: (always 0).

M The major device number.

x The number of the host adapter and if necessary the numbers of the SCSI bus on the hardware [the number coding depends on the machine, for more information see sdisk(7)].

d The SCSI tape drive number.

u Unused.

v Encodes the variable block mode. This bit is used for tapes which may be used in fixed and variable block mode. When 0, fixed block mode is used; when 1, variable block mode is used.

g The code that defines whether the generic interface for reading and writing by means of ioctl system calls may be used for this device. If the bit is not set, each attempt to write to the device by means of ioctl system calls is returned with an error message. The *g* bit cannot be combined with other bits. The meaning is different when combined with the *n* bit (see description of the *n* bit).

C Encodes the compression density attribute, which is currently only used for Exabyte 8mm video tapes with compression modes. When 1, the device is selected as the compression density/speed unit. When 0, the device's default density/speed is used. As bit *h*, this bit is ignored during read operations.

n Encodes the no-rewind attribute. When 1, the tape is positioned at the End-of-Media (EOM) side of the filemark at close. When 0, the tape is rewound to Beginning-of-Media at close. Refer to the mtio(7) manual page for more detailed information. If the *g* bit is set in addition to the *n* bit, the tape remains at its current position when `close` is invoked (no wind forward or rewind operations).

h Encodes the density attribute. When 1, the device is selected as the high density/speed unit. When 0, the device's default density/speed is used. This bit is ignored during read operations.

Major-minor device numbers are assigned dynamically for SCSI devices on routers and for SCSI LUNs. In this case, the major-minor device numbers have no decodable format.

ERROR DESCRIPTION

The error messages displayed for the SCSI tape drives have the following format:

```
ios0/stape002: <02090200> Unit Not Ready - No Cartridge
```

where:

`ios0/stape002:`
The SCSI tape drive that had an error. In this case, the tape drive located at device address 2.

<*KKAASSCC*>
The SCSI composite status code. The composite status code has the following format:

KK The SCSI Sense Key value.

AA The SCSI Additional Sense Key value.

SS The SCSI status returned to the SCSI host adapter.

CC The SCSI host adapter error code.

`Unit Not Ready`
The decoded SCSI Sense Key message.

`No Cartridge`
The decoded SCSI Additional Sense Key message.

SCSI TAPE CONFIGURATION

Device nodes are created using the `/sbin/tpsetup` command. Refer to the tpsetup(8) manual page for more information.

SCSI tape devices are identified during boot. The tape type is determined by issuing the SCSI inquiry command. Bytes 8 to 31, the Vendor and Product identification, returned by the SCSI inquiry are matched against the inquiry names kept by the SCSI tape profile tables defined in `/etc/conf/pack.d/sti/space.c`. The name in `space.c` must match exactly, byte for byte, otherwise the tape will be classified as "unknown".

The command autoconf(8) can be used to determine if a tape drive is recognized by the system at boot. For example, using `autoconf -l`:

```
ios0/stape000 Tandberg TDC 3600
```

appears for a Tandberg QIC drive recognized by the system.

```
ios0/stape000: not configured - unknown type
```

appears for a tape drive the system did not recognize. This drive would be classified as "unknown" by the driver.

An "unknown" tape drive is any drive that does not have a tape profile defined. Default parameters are used to allow the drive to function. However, since a minimal error table is defined to handle tape positioning status (end-of-tape and filemarks), error messages can not be decoded from the composite status word. In addition, some tape drives require specific bits to be set in order to function correctly. If those tape drives are not recognized and are classified as "Generic", they may not function properly. It is suggested that all tape drives are configured onto the system using the appropriate tables so the system recognizes the drive correctly at boot.

Density selection occurs on every tape open with a SCSI mode select command. Other drive specific

parameters are also set during tape open. These parameters are specified by the default mode select parameter entry for the tape type in `/etc/conf/pack.d/sti/space.c`.

GENERAL INFORMATION ON SCSI TAPE DEVICES

Aborting jobs

When a job is issued for a tape device, it returns after a certain period of time. The length of time that elapses depends on the type of job (it can take several hours for a complete physical delete job) and the type of tape device. If a job, which normally returns after a few seconds, does not return after several minutes, you can assume that a problem has arisen. The user has the option from Reliant UNIX V5.44 onwards of forcibly terminating such jobs using the `SIGKILL` signal or the `mt` command with the `reset` parameter. The operating system itself also identifies that a problem has arisen, but only aborts the job after a specified period has elapsed (timeout). The timeout can be as long as six hours. If a tape job is aborted using the `SIGKILL` (`kill -9`) signal or the `mt -f /dev/ios`*X*`/rstape`*XXX* `reset` command, the data written to tape up to this point can no longer be used and the last job must be repeated.

Monitoring cleaning cycles

All tape drives must be cleaned after a certain number of jobs. Recommendations have been given on how often certain tape drives should be cleaned depending on operating conditions. From Reliant UNIX V5.44 onwards, the operating system automatically monitors whether or not any of the connected tape drives need to be cleaned. If so, the following message is output on the system console:

```
CMN: NOTICE: sti 787 ios0/stapeXXX It's time to clean this drive
```

This message is output the first time the operating system determines that the drive needs to be cleaned. If the drive continues to be used, the following warning is output on the system console:

```
CMN: NOTICE: sti 790 Please clean the drive: ios0/stapeXXX
```

If you continue to use the drive without cleaning it, the following message is output on the system console after, for instance, the data on the data medium has been written or read:

```
CMN: NOTICE: sti 788 ios0/stapeXXX must be cleaned before use
```

If the tape drive is to continue being used, it must be cleaned immediately. If it is not cleaned, any further attempts by the operating system to access the drive are rejected. This is a precautionary measure designed to protect the tape drive. You could seriously damage the EXABYTE drives, if you continue to use them without first cleaning them. The following types of tape drive are affected here:

- All 8mm EXABYTE
- All PXT1-B1X (IBM-MAGSTAR)
- All DAT drives.

SCSI TAPE NOTES

When SCSI tape devices are used for either reading or writing, some limitations apply. SCSI tapes may be accessed as raw (character) devices only. Access through the block device will result in `EINVAL` being returned at open.

When reading tape devices, the buffer start address must be "word aligned".

Depending on the version of the operating system, the following values for maximum block size apply:

Operating system version		
RM400	V5.41	128-2
RM600	V5.41 B	64
RM400, RM600	V5.42	128
RM600	V5.43	512
RM600 E models	V5.44	1024

Note for RM600:

Block sizes over 64 KB are only supported if their length is "long word aligned". Other length specifications are rejected.

Reading and writing blocks of any arbitrary size within this limit is allowed if the unit supports variable blocked mode.

The tape position can only be determined by moving the tape. The tape position status can only be obtained through the extended sense bytes returned by the SCSI request sense command after a tape movement command. To determine tape position, the driver first issues a backspace record (BSR) `ioctl`. There can only be three outcomes for this movement. The first possibility is the BSR reported a beginning-of-tape (BOT) status. In this case, no further action is required since a BSR at BOT leaves the tape at BOT. The second possibility is that a filemark was encountered during the BSR. The tape position was thus at the EOM side of a filemark, so in order to restore the tape position, a forward-space-file (FSF) is issued. The final possibility is that the BSR completed without any error. This case requires a forward-space-record (FSR) to restore the tape position.

Some SCSI tape drives return a "SCSI Busy" status when the drive is still busy with a previous command (e.g. rewinding). The driver will retry the I/O until either the device completes the I/O without returning SCSI Busy status or the "Busy Retry Count" (as specified in the tape's profile structure) is exhausted. The I/O will be retried at the interval defined by the "Busy Retry Interval Time" in the tape's profile structure.

QIC CARTRIDGE TAPE NOTES

Quarter-Inch Compatible (QIC) cartridge tape drives have different tape capacities. Interchangeability is preserved since all QIC drives can always read tapes written by other lower capacity tape drives. So, for instance, a QIC tape written using the 60-Mbyte tape QIC format can be read by a drive that supports the 150-Mbyte QIC format (the reverse situation would, of course, not be true).

Depending on the cassette drive, either fixed block mode only or variable *and* fixed block mode is supported in the case of QIC devices.

In the case of tapes with fixed block size, where the minimum block size is lower than the fixed block size, the data transferred is filled with zeros when writing until the fixed block size is reached. When reading, the block is read completely from the tape, but only the required number of bytes are transferred for the call.

A 150-Mbyte tape is only written with 150 Mbytes of data when a high-density device is used, otherwise low density (120-Mbyte) is used.

Note:

The maximum block size in variable block mode in QIC-150 format is 32 KB.

In a 525-Mbyte CARTRIDGE DRIVE, 525 Mbytes are always recorded when 525-Mbyte tapes are used (irrespective of whether they are low- or high-density devices). A 150-Mbyte tape in a 525-Mbyte drive is always written with 150 Mbytes.

There is also the exception in that tapes that have already been written with 120 Mbytes, are recorded again with 120 Mbytes.

For QIC drives, only one filemark is written on tape closure. Normally two filemarks are written whenever a tape is closed, and the tape is positioned between the two filemarks. Writing the next file would overwrite the second filemark with data. QIC drives cannot overwrite filemarks, so the second filemark is omitted. The tape position is the same however, on the EOM side of a filemark (the second filemark does not exist).

Data cannot be added at the end of a QIC tape unless the `MTAPP` operation was used to position the tape at the end of the data area [see `mtio(7)`]. A series of files may be written to QIC tape by issuing successive writes to the no-rewind device. However, once the last file has been written and the tape rewound or repositioned in any way, attempts to position the tape at the end of the last file and write additional data will result in an error.

QIC tape cartridges have stringent maintenance recommendations. The recommended tape cleaning interval is every eight hours of use, with the supplied tape cleaning cartridge. Besides maintenance, it is recommended that all tapes be retensioned upon tape insertion. As shipped, auto-retensioning is disabled in

the default mode select parameters for QIC tapes in `/etc/conf/pack.d/sti/space.c`. This option can be enforced by changing the default mode select parameters for the tape type. The major impact of auto-retensioning is that tapes are positioned to EOM, and then back to BOT. This can take three minutes or more. The auto-retensioning time can be halved if the tape drive supports moving the tape to EOM before unloading the cartridge.

NINE-TRACK TAPE NOTES

Nine-track tapes are considered variable-mode devices (blocks up to 128 KB are possible). Although nine-track drives usually support either variable- or fixed-block modes, the recommended usage for the drive is variable mode. Support for nine-track tape drives is limited to only the SCSI differential electrical interface.

DAT/DDS NOTES

Digital Audio Tape (DAT) drive behavior is similar to nine-track tapes. Although DAT drives support either variable- or fixed-block modes, they are considered variable-mode devices. It is recommended that they be used in variable-block mode.

The DAT drives supported are industry standard Digital Data Storage (DDS) format. Tape interchange is possible with other standard DDS DAT drives. However, tapes written in DDS tape drives can NOT be read in incompatible non-DDS DAT (such as DATA/DAT) drives (and vice versa: tapes written in non-DDS DAT drives can NOT be read in the DDS drive).

The tapes used in the DDS drive are industry standard DAT tapes. It is strongly recommended that only DATA QUALITY tapes are used in the drive. Audio quality DAT tapes may not be as reliable. The tapes are available in several different lengths.

DDS-1 60m
: 1.3 GB

DDS-2 90m
: 2.0 GB

DDS-2 120m
: 4.0 GB (8.0 GB in compressed mode only)

DDS-3 125m
: 12.0 GB (24.0 GB in compressed mode only)

formats. The format in which a tape can be written can be ascertained from the directory entry.

Format DDS-1
: `/dev/ios`*b*`/rstape`*ppu*

Format DDS-2
: `/dev/ios`*b*`/rstape`*ppu*`h`

Format DDS-2 (compressed)
: `/dev/ios`*b*`/rstape`*ppu*`ch`

Format DDS-3
: `/dev/ios`*b*`/rstape`*ppu*`h`

Format DDS-3 (compressed)
: `/dev/ios`*b*`/rstape`*ppu*`ch`

Note: DSS-1 tapes cannot be written in DSS-2 format nor can DSS-2 tapes be written in DSS-3 format.

The exact amount of data stored on the tape may vary a little depending on the number of retries needed to write the data to tape and if the drive was streaming.

Tips for avoiding data backup problems

Remove all sources of dust and dirt from the vicinity of systems with data recorders. Paper and toner dust from printers and photocopiers always pose a danger.

The data recorder should only ever be used in streaming mode. The wear and tear on the head drum and the

magnetic tapes is increased enormously in start/stop mode. Streaming mode can be achieved by

- recording the data to be backed up in blocks ≥ 16 Kbytes
- performing backups at times when the system utilization is low

Do not leave the data cartridge in the data recorder for longer than absolutely necessary. The safest place for the cartridge is the anti-static, dust-proof storage case supplied.

If the cartridges were subjected to climatic conditions other than the operating environment, allow the cartridges at least 24 hours to acclimatize before using them.

Store the data cartridges in an environment with a constant temperature and humidity. The optimum temperature is 5°C - 32°C and 20% – 60% rel. humidity. Never allow a data cartridge to be exposed to direct sunlight or to electrical or magnetic fields.

Cleaning

The data recorder must be cleaned at regular intervals. If it is not cleaned at all, or only inadequately, the read/write error rate will increase, the actual tape will wear out more quickly and particles of dirt in the head drum may begin to smolder. In this last case, it is already too late to use a cleaning cartridge. The drive will have to be sent to the Service Center for cleaning and checking.

Cleaning Cartridges can be ordered from SIEMENS NIXDORF PLUS.

Cleaning intervals:

- daily, if you use the drive four to five times per day
- twice a week, if you use the drive between two and three times per day
- weekly, if you use the drive briefly each day

If the system environment is dusty, as is often the case in offices, much shorter intervals between cleaning will be required.

If the error display flashes in amber, the drive should be cleaned immediately. This is always if the tape pulling hours is more than 20 hours, or the error rate is to high or "`MEDIUM ERROR`" occurs.

Please observe the following cleaning procedure:

- Load the cleaning cartridge
- The cleaning procedure starts automatically; it lasts 30 seconds
- After the cleaning procedure, the cleaning cartridge will be ejected automatically
- Remove the cleaning cartridge.
- Note down the date of cleaning on the cartridge label provided for this purpose.

You can use the cleaning cartridge approximately 40 times.

Data Cartridges

Only use 4mm magnetic tape cassettes with the "DDS" (Digital Data Storage) and "MRS IIII" (Media Recognition System) symbols, as other cassettes can cause a malfunction. Suitable 4mm cassettes are listed under "Accessories" in the appendix of your Operating Manual. DAT audio cassettes cannot be written or read.

Data cartridges are subject to wear and tear, and in optimum operating conditions can be used up to 500 times. Poor operating conditions (e.g. dusty environment, start/stop mode) will reduce the cartridge lifetime to 30 operation cycles.

Replace the data cartridge immediately with a new cartridge, if during data storage "`MEDIUM ERROR`" occurs.

NOTES ON EXABYTE 8MM DATA RECORDERS

Exabyte devices can write or read 8mm data tapes in four logical formats. The format in which a tape can be written can be ascertained from the directory entry.

EXB-8900c	compressed format	`/dev/ios`*b*`/rstape`*pp* *u*`ch`

EXB-8900	uncompressed format	/dev/ios*b*/rstape*pp**u*h
EXB-8500c	compressed format	/dev/ios*b*/rstape*pp**u*ch
EXB-8500	uncompressed format	/dev/ios*b*/rstape*pp**u*c
EXB-8200c	compressed format	/dev/ios*b*/rstape*pp**u*c
EXB-8200	uncompressed format	/dev/ios*b*/rstape*pp**u*

Important:

Not all Exabyte devices support all formats. For example, the compressed format is only supported by Exabyte devices which have a compress facility.

Device	Format
MC75 EXB-8900	8900c, 8900 (read/write) 8500c, 8500, 8200 (readonly)
MC15 ELIANT 820	8500c, 8500 (read/write), 8200 (readonly)
MC10 EXB-8505	8500c, 8500, 8200
MC10 EXB-8500	8500, 8200
MC10 EXB-8200	8200

Tips for avoiding data backup problems

Remove all sources of dust and dirt from the vicinity of systems with data recorders. Paper and toner dust from printers and photocopiers always pose a danger.

The data recorder should only ever be used in streaming mode. The wear and tear on the head drum and the magnetic tapes is increased enormously in start/stop mode. Streaming mode can be achieved by

- recording the data to be backed up in blocks ≥ 16 Kbytes
- performing backups at times when the system utilization is low

Do not leave the data cartridge in the data recorder for longer than absolutely necessary. The safest place for the cartridge is the anti-static, dust-proof storage case supplied.

If the cartridges were subjected to climatic conditions other than the operating environment, allow the cartridges at least 24 hours to acclimatize before using them.

Store the data cartridges in an environment with a constant temperature and humidity. The optimum temperature is 20°C ±2°C and 40% – 60% rel. humidity. Never allow a data cartridge to be exposed to direct sunlight or to electrical or magnetic fields.

Cleaning

The data recorder must be cleaned at regular intervals. If it is not cleaned at all, or only inadequately, the read/write error rate will increase, the actual tape will wear out more quickly and particles of dirt in the head drum may begin to smolder. In this last case, it is already too late to use a cleaning cartridge. The drive will

have to be sent to the Service Center for cleaning and checking.

There are many cleaning procedures, most of which originate in Video Tape technology (e.g. cleaning by means of solutions). Many procedures are not suitable for cleaning data recorders. Using cleaning procedures of this kind can considerably impair the functioning of the device and even damage it permanently. You should only use the cleaning cartridges approved by SIEMENS NIXDORF and EXABYTE.

Only use one of the following cartridges types:

- EXABYTE 8mm Cleaning Cartridge
- SONY QG6-CL

Cleaning Cartridges can be ordered from SIEMENS NIXDORF PLUS.

You should clean the data recorder at least once per month or after a maximum of 30 hours of operating time, whichever is reached first.

If the system environment is dusty, as is often the case in offices, much shorter intervals between cleaning will be required.

Please observe the following cleaning procedure:

- Load the cleaning cartridge.
- The cleaning procedure starts automatically and lasts 30 seconds to 4 minutes, depending of the type of drive
- After the cleaning procedure, the cleaning cartridge will be ejected automatically.
- Remove the cleaning cartridge.
- Note down the date of cleaning on the cartridge label provided for this purpose.

A special command needs to be executed to prepare the cleaning procedure with operating system versions V5.42 and V5.43. Please execute the command `clean8mm` before beginning the cleaning procedure described above. Follow the instructions that appear on the console.

You will find further information in `clean8mm(8)`.

Depending on the cleaning cartridge type, 3 or 12 cleaning procedures can be carried out. You should never tamper with the cleaning cartridge by trying to rewind it.

Data Cartridges

You should only use data cartridges approved by SIEMENS NIXDORF and EXABYTE. You can only ensure that the type and number of tape errors will be within a permissible range by using these cartridges. The magnetic tape has an anti-static coating and is carefully designed for processing digital data. Friction on coating particles and the associated wear and tear on the head drum are reduced to a minimum.

Only use one of the following cartridges types:

- EXATAPE AME (8900 only)
- EXATAPE
- SONY QG112MA

The SONY QG112MA cartridge is available from SIEMENS NIXDORF PLUS.

The EXABYTE AME tape must be used for the EXB-8900 device. Older tapes can be read (however, this should remain the exception). The drive must first be cleaned before you next write to AME tapes.

Data cartridges are subject to wear and tear process, and in optimum operating conditions can be used up to 500 times. Poor operating conditions (e.g. dusty environment, start/stop mode) reduce the cartridge lifetime to 30 operation cycles.

Replace the data cartridge immediately with a new cartridge, if during data storage "`MEDIUM ERROR`" or "`EXCESSIVE WRITE ERROR`" occurs.

FILES

`/dev/ios`*b*`/[r]stape`*ppu*`[c][h][n][o][v][g]`

/dev/ios*b*/[r]stape*ppu*.*l*[c][h][n][o][v][g]

/dev/iof0/[r]stape.*pp*.*nnnnn*.*ccu*.*l*[c][h][n][o][v][g]

/devices/[r]*serialnumber*.[c][h][n][o][v][g]

/etc/conf/pack.d/sti/space.c

On RM400 servers, the tape devices are also responsive as

/dev/rmt/ctape[0|4] (high density)

/dev/rmt/vtape[0|4] (no-rewind)

SEE ALSO

cpio(1), mt(1), tar(1), devname(7), mtio(7), sdisk(7), clean8mm(8), cleandev(8), tpsetup(8).

System Administrator's Reference Manual

streamio(7)

NAME

`streamio` - STREAMS ioctl commands

SYNOPSIS

```
#include <sys/types.h>
#include <stropts.h>

int ioctl(int fildes, int command, ... /* arg */);
```

DESCRIPTION

STREAMS `ioctl` commands are a subset of the `ioctl(2)` system calls which perform a variety of control functions on streams.

fildes is an open file descriptor that refers to a stream. *command* determines the control function to be performed as described below. *arg* represents additional information that is needed by this command. The type of *arg* depends upon the command, but it is generally an integer or a pointer to a *command*-specific data structure. The *command* and *arg* are interpreted by the stream head. Certain combinations of these arguments may be passed to a module or driver in the stream.

Since these STREAMS commands are a subset of `ioctl`, they are subject to the errors described there. In addition to those errors, the call will fail with `errno` set to `EINVAL`, without processing a control function, if the stream referenced by *fildes* is linked below a multiplexer, or if *command* is not a valid value for a stream.

Also, as described in `ioctl`, STREAMS modules and drivers can detect errors. In this case, the module or driver sends an error message to the stream head containing an error value. This causes subsequent system calls to fail with `errno` set to this value.

COMMAND FUNCTIONS

The following `ioctl` commands, with error values indicated, are applicable to all STREAMS files:

`I_PUSH`

Pushes the module whose name is pointed to by *arg* onto the top of the current stream, just below the stream head. If the stream is a pipe, the module will be inserted between the stream heads of both ends of the pipe. It then calls the open routine of the newly-pushed module. On failure, `errno` is set to one of the following values:

`EINVAL`

Invalid module name.

`EFAULT`

arg points outside the allocated address space.

`ENXIO`

Open routine of new module failed.

`ENXIO`

Hangup received on *fildes*.

`I_POP`

Removes the module just below the stream head of the stream pointed to by *fildes*. To remove a module from a pipe requires that the module was pushed on the side it is being removed from. *arg* should be 0 in an `I_POP` request. On failure, `errno` is set to one of the following values:

`EINVAL`

No module present in the stream.

`ENXIO`

Hangup received on *fildes*.

I_LOOK
Retrieves the name of the module just below the stream head of the stream pointed to by *fildes*, and places it in a null terminated character string pointed at by *arg*. The buffer pointed to by *arg* should be at least FMNAMESZ+1 bytes long. An (#include <sys/conf.h>) declaration is required. On failure, errno is set to one of the following values:

EFAULT
arg points outside the allocated address space.

EINVAL
No module present in stream.

I_FLUSH
This request flushes all input and/or output queues, depending on the value of *arg*. Legal *arg* values are:

FLUSHR
Flush read queues.

FLUSHW
Flush write queues.

FLUSHRW
Flush read and write queues.

If a pipe or FIFO does not have any modules pushed, the read queue of the stream head on either end is flushed depending on the value of *arg*.

If FLUSHR is set and *fildes* is a pipe, the read queue for that end of the pipe is flushed and the write queue for the other end is flushed. If *fildes* is a FIFO, both queues are flushed.

If FLUSHW is set and *fildes* is a pipe and the other end of the pipe exists, the read queue for the other end of the pipe is flushed and the write queue for this end is flushed. If *fildes* is a FIFO, both queues of the FIFO are flushed.

If FLUSHRW is set, all read queues are flushed, that is, the read queue for the FIFO and the read queue on both ends of the pipe are flushed.

Correct flush handling of a pipe or FIFO with modules pushed is achieved via the pipemod module. This module should be the first module pushed onto a pipe so that it is at the midpoint of the pipe itself.

On failure, errno is set to one of the following values:

ENOSR
Unable to allocate buffers for flush message due to insufficient STREAMS memory resources.

EINVAL
Invalid *arg* value.

ENXIO
Hangup received on *fildes*.

I_FLUSHBAND
Flushes a particular band of messages. *arg* points to a bandinfo structure that has the following members:

```
unsigned char   bi_pri;
int             bi_flag;
```

The bi_flag field may be one of FLUSHR, FLUSHW, or FLUSHRW as described earlier.

I_SETSIG
Informs the stream head that the user wishes the kernel to issue the SIGPOLL signal [see signal(2)] when a particular event has occurred on the stream associated with *fildes*. I_SETSIG supports an asynchronous processing capability in STREAMS. The value of *arg* is a bitmask that specifies the events for which the user should be signaled. It is the bitwise-OR of any combination of the following constants:

S_INPUT
: Any message other than an M_PCPROTO has arrived in a stream head read queue. This event is maintained for compatibility with prior UNIX System V releases. This is set even if the message is of zero length.

S_RDNORM
: An ordinary (non-priority) message has arrived in a stream head read queue. This is set even if the message is of zero length.

S_RDBAND
: A priority band message (band > 0) has arrived in a stream head read queue. This is set even if the message is of zero length.

S_HIPRI
: A high priority message is present in the stream head read queue. This is set even if the message is of zero length.

S_OUTPUT
: The write queue just below the stream head is no longer full. This notifies the user that there is room in the queue for sending (or writing) data downstream.

S_WRNORM
: This event is the same as S_OUTPUT.

S_WRBAND
: A priority band greater than 0 of a queue downstream exists and is writable. This notifies the user that there is room in the queue for sending (or writing) priority data downstream.

S_MSG
: ASTREAMS signal message that contains the SIGPOLL signal has reached the front of the stream head read queue.

S_ERROR
: An M_ERROR message has reached the stream head.

S_HANGUP
: An M_HANGUP message has reached the stream head.

S_BANDURG
: When used in conjunction with S_RDBAND, SIGURG is generated instead of SIGPOLL when a priority message reaches the front of the stream head read queue.

A user process may choose to be signaled only of high priority messages by setting the *arg* bitmask to the value S_HIPRI.

Processes that wish to receive SIGPOLL signals must explicitly register to receive them using I_SETSIG. If several processes register to receive this signal for the same event on the same stream, each process will be signaled when the event occurs.

If the value of *arg* is zero, the calling process will be unregistered and will not receive further SIGPOLL signals. On failure, errno is set to one of the following values:

EINVAL
: *arg* value is invalid or *arg* is zero and process is not registered to receive the SIGPOLL signal.

EAGAIN
: Allocation of a data structure to store the signal request failed.

I_GETSIG
: Returns the events for which the calling process is currently registered to be sent a SIGPOLL signal. The events are returned as a bitmask pointed to by *arg*, where the events are those specified in the description of I_SETSIG above. On failure, errno is set to one of the following values:

EINVAL
: Process not registered to receive the SIGPOLL signal.

EFAULT
: *arg* points outside the allocated address space.

I_FIND
: Compares the names of all modules currently present in the stream to the name pointed to by *arg*, and returns 1 if the named module is present in the stream. It returns 0 if the named module is not present. On failure, errno is set to one of the following values:

  EFAULT
  : *arg* points outside the allocated address space.

  EINVAL
  : *arg* does not contain a valid module name.

I_PEEK
: Allows a user to retrieve the information in the first message in the stream head read queue without taking the message off the queue. I_PEEK is analogous to getmsg(2) except that it does not remove the message from the queue. *arg* points to a strpeek structure which contains the following members:

```
struct strbuf   ctlbuf;
struct strbuf   databuf;
long    flags;
```

  The maxlen field in the ctlbuf and databuf strbuf structures [see getmsg(2)] must be set to the number of bytes of control information and/or data information, respectively, to retrieve. *flags* may be set to RS_HIPRI or 0. If RS_HIPRI is set, I_PEEK will look for a high priority message in the stream head read queue. Otherwise, I_PEEK will look for the first message on the stream head read queue.

  I_PEEK returns 1 if a message was retrieved, and returns 0 if no message was found in the stream head read queue. It does not wait for a message to arrive. On return, ctlbuf specifies information in the control buffer, *databuf* specifies information in the data buffer, and *flags* contains the value RS_HIPRI or 0. On failure, errno is set to the following value:

  EFAULT
  : *arg* points, or the buffer area specified in *ctlbuf* or *databuf* is, outside the allocated address space.

  EBADMSG
  : Queued message to be read is not valid for I_PEEK.

  EINVAL
  : Illegal value for *flags*.

I_SRDOPT
: Sets the read mode [see read(2)] using the value of the argument *arg*. Legal *arg* values are:

  RNORM
  : Byte-stream mode, the default.

  RMSGD
  : Message-discard mode.

  RMSGN
  : Message-nondiscard mode.

  In addition, treatment of control messages by the stream head may be changed by setting the following flags in *arg*:

  RPROTNORM
  : Fail read() with EBADMSG if a control message is at the front of the stream head read queue.

  RPROTDAT
  : Deliver the control portion of a message as data when a user issues read(). This is the default behavior.

  RPROTDIS
  : Discard the control portion of a message, delivering any data portion, when a user issues a read().

  On failure, errno is set to the following value:

EINVAL
arg is not one of the above legal values.

I_GRDOPT
Returns the current read mode setting in an `int` pointed to by the argument *arg*. Read modes are described in `read(2)` . On failure, `errno` is set to the following value:

EFAULT
arg points outside the allocated address space.

I_NREAD
Counts the number of data bytes in data blocks in the first message in the stream head read queue, and places this value in the location pointed to by *arg*. The return value for the command is the number of messages in the stream head read queue. For example, if zero is returned in *arg*, but the `ioctl` return value is greater than zero, this indicates that a zero-length message is next in the queue. On failure, `errno` is set to the following value:

EFAULT
arg points outside the allocated address space.

I_FDINSERT
Creates a message from user specified buffer(s), adds information about another stream and sends the message downstream. The message contains a control part and an optional data part. The data and control parts to be sent are distinguished by placement in separate buffers, as described below.

arg points to a `strfdinsert` structure which contains the following members:

```
struct strbuf   ctlbuf;
struct strbuf   databuf;
long    flags;
int     fildes;
int     offset;
```

The `len` field in the `ctlbuf strbuf` structure [see `putmsg(2)`] must be set to the size of a pointer plus the number of bytes of control information to be sent with the message. *fildes* in the `strfdinsert` structure specifies the file descriptor of the other stream. *offset*, which must be word-aligned, specifies the number of bytes beyond the beginning of the control buffer where `I_FDINSERT` will store a pointer. This pointer will be the address of the read queue structure of the driver for the stream corresponding to *fildes* in the `strfdinsert` structure. The `len` field in the `databuf strbuf` structure must be set to the number of bytes of data information to be sent with the message or zero if no data part is to be sent.

flags specifies the type of message to be created. An ordinary (non-priority) message is created if *flags* is set to 0, a high priority message is created if *flags* is set to `RS_HIPRI`. For normal messages, `I_FDINSERT` will block if the stream write queue is full due to internal flow control conditions. For high priority messages, `I_FDINSERT` does not block on this condition. For normal messages, `I_FDINSERT` does not block when the write queue is full and `O_NDELAY` or `O_NONBLOCK` is set. Instead, it fails and sets `errno` to `EAGAIN`.

`I_FDINSERT` also blocks, unless prevented by lack of internal resources, waiting for the availability of message blocks, regardless of priority or whether `O_NDELAY` or `O_NONBLOCK` has been specified. No partial message is sent. On failure, `errno` is set to one of the following values:

EAGAIN
A non-priority message was specified, the `O_NDELAY` or `O_NONBLOCK` flag is set, and the stream write queue is full due to internal flow control conditions.

ENOSR
Buffers could not be allocated for the message that was to be created due to insufficient STREAMS memory resources.

EFAULT
arg points, or the buffer area specified in *ctlbuf* or *databuf* is, outside the allocated address

space.

EINVAL
: One of the following: *fildes* in the `strfdinsert` structure is not a valid, open stream file descriptor; the size of a pointer plus *offset* is greater than the `len` field for the buffer specified through *ctlptr*; *offset* does not specify a properly-aligned location in the data buffer; an undefined value is stored in *flags*.

ENXIO
: Hangup received on *fildes* of the `ioctl` call or *fildes* in the `strfdinsert` structure.

ERANGE
: The `len` field for the buffer specified through *databuf* does not fall within the range specified by the maximum and minimum packet sizes of the topmost stream module, or the `len` field for the buffer specified through *databuf* is larger than the maximum configured size of the data part of a message, or the `len` field for the buffer specified through *ctlbuf* is larger than the maximum configured size of the control part of a message.

`I_FDINSERT` can also fail if an error message was received by the stream head of the stream corresponding to *fildes* in the `strfdinsert` structure. In this case, `errno` will be set to the value in the message.

`I_STR`
: Constructs an internal STREAMS `ioctl` message from the data pointed to by *arg*, and sends that message downstream.

This mechanism is provided to send user `ioctl` requests to downstream modules and drivers. It allows information to be sent with the `ioctl`, and will return to the user any information sent upstream by the downstream recipient. `I_STR` blocks until the system responds with either a positive or negative acknowledgement message, or until the request "times out" after some period of time. If the request times out, it fails with `errno` set to `ETIME`.

At most, one `I_STR` can be active on a stream. Further `I_STR` calls will block until the active `I_STR` completes at the stream head. The default timeout interval for these requests is 15 seconds. The `O_NDELAY` and `O_NONBLOCK` [see open(2)] flags have no effect on this call.

To send requests downstream, *arg* must point to a `strioctl` structure which contains the following members:

```
int     ic_cmd;
int     ic_timout;
int     ic_len;
char    *ic_dp;
```

`ic_cmd` is the internal `ioctl` command intended for a downstream module or driver and `ic_timout` is the number of seconds (−1 = infinite, 0 = use default, > 0 = as specified) an `I_STR` request will wait for acknowledgement before timing out. The default timeout is infinite. `ic_len` is the number of bytes in the data argument and `ic_dp` is a pointer to the data argument. The `ic_len` field has two uses: on input, it contains the length of the data argument passed in, and on return from the command, it contains the number of bytes being returned to the user (the buffer pointed to by `ic_dp` should be large enough to contain the maximum amount of data that any module or the driver in the stream can return).

The stream head will convert the information pointed to by the `strioctl` structure to an internal `ioctl` command message and send it downstream. On failure, `errno` is set to one of the following values:

ENOSR
: Unable to allocate buffers for the `ioctl` message due to insufficient STREAMS memory resources.

EFAULT
: *arg* points, or the buffer area specified by `ic_dp` and `ic_len` (separately for data sent and data returned) is, outside the allocated address space.

EINVAL
ic_len is less than 0 or ic_len is larger than the maximum configured size of the data part of a message or ic_timout is less than –1.

ENXIO
Hangup received on *fildes*.

ETIME
A downstream ioctl timed out before acknowledgement was received.

An I_STR can also fail while waiting for an acknowledgement if a message indicating an error or a hangup is received at the stream head. In addition, an error code can be returned in the positive or negative acknowledgement message, in the event the ioctl command sent downstream fails. For these cases, I_STR will fail with errno set to the value in the message.

I_SWROPT
Sets the write mode using the value of the argument *arg*. Legal bit settings for *arg* are:

SNDZERO
Send a zero-length message downstream when a write of 0 bytes occurs.

To not send a zero-length message when a write of 0 bytes occurs, this bit must not be set in *arg*.

On failure, errno may be set to the following value:

EINVAL
arg is the the above legal value.

I_GWROPT
Returns the current write mode setting, as described above, in the int that is pointed to by the argument *arg*.

I_SENDFD
Requests the stream associated with *fildes* to send a message, containing a file pointer, to the stream head at the other end of a stream pipe. The file pointer corresponds to *arg*, which must be an open file descriptor.

I_SENDFD converts *arg* into the corresponding system file pointer. It allocates a message block and inserts the file pointer in the block. The user id and group id associated with the sending process are also inserted. This message is placed directly in the read queue of the stream head at the other end of the stream pipe to which it is connected. On failure, errno is set to one of the following values:

EAGAIN
The sending stream is unable to allocate a message block to contain the file pointer.

EAGAIN
The read queue of the receiving stream head is full and cannot accept the message sent by I_SENDFD.

EBADF
arg is not a valid, open file descriptor.

EINVAL
fildes is not connected to a stream pipe.

ENXIO
Hangup received on *fildes*.

I_RECVFD
Retrieves the file descriptor associated with the message sent by an I_SENDFD ioctl over a stream pipe. *arg* is a pointer to a data buffer large enough to hold an strrecvfd data structure containing the following members:

```
int fd;
uid_t uid;
gid_t gid;
char fill[8];
```

fd is an integer file descriptor. *uid* and *gid* are the user id and group id, respectively, of the sending stream.

If O_NDELAY and O_NONBLOCK are clear [see open(2)], I_RECVFD will block until a message is present at the stream head. If O_NDELAY or O_NONBLOCK is set, I_RECVFD will fail with errno set to EAGAIN if no message is present at the stream head.

If the message at the stream head is a message sent by an I_SENDFD, a new user file descriptor is allocated for the file pointer contained in the message. The new file descriptor is placed in the fd field of the strrecvfd structure. The structure is copied into the user data buffer pointed to by *arg*. On failure, errno is set to one of the following values:

EAGAIN
: A message is not present at the stream head read queue, and the O_NDELAY or O_NONBLOCK flag is set.

EBADMSG
: The message at the stream head read queue is not a message containing a passed file descriptor.

EFAULT
: *arg* points outside the allocated address space.

EMFILE
: NOFILES file descriptors are currently open.

ENXIO
: Hangup received on *fildes*.

EOVERFLOW
: *uid* or *gid* is too large to be stored in the structure pointed to by *arg*.

I_LIST
: Allows the user to list all the module names on the stream, up to and including the topmost driver name. If *arg* is NULL, the return value is the number of modules, including the driver, that are on the stream pointed to by *fildes*. This allows the user to allocate enough space for the module names. If *arg* is non-NULL, it should point to an str_list structure that has the following members:

```
int sl_nmods;
struct str_mlist   *sl_modlist;
```

The str_mlist structure has the following member:

```
char l_name[FMNAMESZ+1];
```

sl_nmods indicates the number of entries the user has allocated in the array and on return, sl_modlist contains the list of module names. The return value indicates the number of entries that have been filled in. On failure, errno may be set to one of the following values:

EINVAL
: The sl_nmods member is less than 1

EAGAIN
: Unable to allocate buffers

I_ATMARK
: Allows the user to see if the current message in the stream head read queue is "marked" by some module downstream. *arg* determines how the checking is done when there may be multiple marked messages in the stream head read queue. It may take the following values:

ANYMARK
: Check if the message is marked.

LASTMARK
: Check if the message is the last one marked in the queue.

The return value is 1 if the mark condition is satisfied and 0 otherwise. On failure, errno may be set to the following value:

EINVAL
Invalid *arg* value.

I_CKBAND
Check if the message of a given priority band exists in the stream head read queue. This returns 1 if a message of a given priority exists, or −1 on error. *arg* should be an integer containing the value of the priority band in question. On failure, errno may be set to the following value:

EINVAL
Invalid *arg* value.

I_GETBAND
Returns the priority band of the first message in the stream head read queue in the integer referenced by *arg*. On failure, errno may be set to the following value:

ENODATA
No message in the stream head read queue.

I_CANPUT
Check if a certain band is writable. *arg* is set to the priority band in question. The return value is 0 if the priority band *arg* is flow controlled, 1 if the band is writable, or −1 on error. On failure, errno may be set to the following value:

EINVAL
Invalid *arg* value.

I_SETCLTIME
Allows the user to set the time the stream head will delay when a stream is closing and there is data in the write queues. Before closing each module and driver, the stream head will delay for the specified amount of time to allow the data to drain. If, after the delay, there is still data present, it will be flushed. *arg* is the number of milliseconds to delay, rounded up to the nearest legal value on the system. The default is fifteen seconds. On failure, errno may be set to the following value:

EINVAL
Invalid *arg* value.

I_GETCLTIME
Returns the close time delay in the integer pointed by *arg*.

The following four commands are used for connecting and disconnecting multiplexed STREAMS configurations.

I_LINK
Connects two streams, where *fildes* is the file descriptor of the stream connected to the multiplexing driver, and *arg* is the file descriptor of the stream connected to another driver. The stream designated by *arg* gets connected below the multiplexing driver. I_LINK requires the multiplexing driver to send an acknowledgement message to the stream head regarding the linking operation. This call returns a multiplexer ID number (an identifier used to disconnect the multiplexer, see I_UNLINK) on success, and a −1 on failure. On failure, errno is set to one of the following values:

ENXIO
Hangup received on *fildes*.

ETIME
Time out before acknowledgement message was received at stream head.

EAGAIN
Temporarily unable to allocate storage to perform the I_LINK.

ENOSR
Unable to allocate storage to perform the I_LINK due to insufficient STREAMS memory resources.

EBADF
arg is not a valid, open file descriptor.

EINVAL

fildes stream does not support multiplexing.

EINVAL
arg is not a stream, or is already linked under a multiplexer.

EINVAL
The specified link operation would cause a "cycle" in the resulting configuration; that is, if a given driver is linked into a multiplexing configuration in more than one place.

EINVAL
fildes is the file descriptor of a pipe or FIFO.

An I_LINK can also fail while waiting for the multiplexing driver to acknowledge the link request, if a message indicating an error or a hangup is received at the stream head of *fildes*. In addition, an error code can be returned in the positive or negative acknowledgement message. For these cases, I_LINK will fail with errno set to the value in the message.

I_UNLINK
Disconnects the two streams specified by *fildes* and *arg*. *fildes* is the file descriptor of the stream connected to the multiplexing driver. *arg* is the multiplexer ID number that was returned by the I_LINK. If *arg* is –1, then all streams which were linked to *fildes* are disconnected. As in I_LINK, this command requires the multiplexing driver to acknowledge the unlink. On failure, errno is set to one of the following values:

ENXIO
Hangup received on *fildes*.

ETIME
Time out before acknowledgement message was received at stream head.

ENOSR
Unable to allocate storage to perform the I_UNLINK due to insufficient STREAMS memory resources.

EINVAL
arg is an invalid multiplexer ID number or *fildes* is not the stream on which the \I_LINK that returned *arg* was performed.

EINVAL
fildes is the file descriptor of a pipe or FIFO.

An I_UNLINK can also fail while waiting for the multiplexing driver to acknowledge the link request, if a message indicating an error or a hangup is received at the stream head of *fildes*. In addition, an error code can be returned in the positive or negative acknowledgement message. For these cases, I_UNLINK will fail with errno set to the value in the message.

I_PLINK
Connects two streams, where *fildes* is the file descriptor of the stream connected to the multiplexing driver, and *arg* is the file descriptor of the stream connected to another driver. The stream designated by *arg* is connected via a persistent link below the multiplexing driver. I_PLINK requires the multiplexing driver to send an acknowledgement message to the stream head regarding the linking operation. This call creates a persistent link which can exist even if the file descriptor *fildes* associated with the upper stream to the multiplexing driver is closed. This call returns a multiplexer ID number (an identifier that may be used to disconnect the multiplexer, see I_PUNLINK) on success, and a –1 on failure. On failure, errno may be set to one of the following values:

ENXIO
Hangup received on *fildes*.

ETIME
Time out before acknowledgement message was received at the stream head.

EAGAIN
Unable to allocate STREAMS storage to perform the I_PLINK.

EBADF
arg is not a valid, open file descriptor.

EINVAL
fildes does not support multiplexing.

EINVAL
arg is not a stream or is already linked under a multiplexer.

EINVAL
The specified link operation would cause a "cycle" in the resulting configuration; that is, if a given stream head is linked into a multiplexing configuration in more than one place.

EINVAL
fildes is the file descriptor of a pipe or FIFO.

An I_PLINK can also fail while waiting for the multiplexing driver to acknowledge the link request, if a message indicating an error on a hangup is received at the stream head of *fildes*. In addition, an error code can be returned in the positive or negative acknowledgement message. For these cases, I_PLINK will fail with errno set to the value in the message.

I_PUNLINK
Disconnects the two streams specified by *fildes* and *arg* that are connected with a persistent link. *fildes* is the file descriptor of the stream connected to the multiplexing driver. *arg* is the multiplexer ID number that was returned by I_PLINK when a stream was linked below the multiplexing driver. If *arg* is MUXID_ALL then all streams which are persistent links to *fildes* are disconnected. As in I_PLINK, this command requires the multiplexing driver to acknowledge the unlink. On failure, errno may be set to one of the following values:

ENXIO
Hangup received on *fildes*.

ETIME
Time out before acknowledgement message was received at the stream head.

EAGAIN
Unable to allocate buffers for the acknowledgement message.

EINVAL
Invalid multiplexer ID number.

EINVAL
fildes is the file descriptor of a pipe or FIFO.

An I_PUNLINK can also fail while waiting for the multiplexing driver to acknowledge the link request if a message indicating an error or a hangup is received at the stream head of *fildes*. In addition, an error code can be returned in the positive or negative acknowledgement message. For these cases, I_PUNLINK will fail with errno set to the value in the message.

DIAGNOSTICS

Unless specified otherwise above, the return value from ioctl is 0 upon success and –1 upon failure with errno set as indicated.

SEE ALSO

close(2), fcntl(2), getmsg(2), intro_prm2(2), ioctl(2), open(2), poll(2), putmsg(2), read(2), signal(2), write(2), signal(5).

Programmer's Guide: STREAMS

System Administrator's Reference Manual

sxt(7)

NAME

`sxt` - pseudo-device driver

DESCRIPTION

The special file `/dev/sxt` is a pseudo device driver that interposes a discipline between the standard `tty` line disciplines and a real device driver. The standard disciplines manipulate virtual tty structures (channels) declared by the `/dev/sxt` driver. `/dev/sxt` acts as a discipline manipulating a real tty structure declared by a real device driver. The `/dev/sxt` driver is currently only used by the `shl(1)` command.

Virtual ttys are named by inodes in the subdirectory `/dev/sxt` and are allocated in groups of up to eight. To allocate a group, a program should exclusively open a file with a name of the form `/dev/sxt/??0` (channel 0) and then execute a `SXTIOCLINK ioctl` call to initiate the multiplexing.

Only one channel, the controlling channel, can receive input from the keyboard at a time; others attempting to read will be blocked.

There are two groups of `ioctl(2)` commands supported by `sxt`. The first group contains the standard `ioctl` commands described in `termio(7)`, with the addition of the following:

`TIOCEXCL`
: Set exclusive use mode: no further opens are permitted until the file has been closed.

`TIOCNXCL`
: Reset exclusive use mode: further opens are once again permitted.

The second group are commands to `sxt` itself. Some of these may only be executed on channel 0.

`SXTIOCLINK`
: Allocate a channel group and multiplex the virtual ttys onto the real tty. The argument is the number of channels to allocate. This command may only be executed on channel 0. Possible errors include:

  `EINVAL`
  : The argument is out of range.

  `ENOTTY`
  : The command was not issued from a real tty.

  `ENXIO`
  : `linesw` is not configured with `sxt`.

  `EBUSY`
  : An `SXTIOCLINK` command has already been issued for this real `tty`.

  `ENOMEM`
  : There is no system memory available for allocating the virtual tty structures.

  `EBADF`
  : Channel 0 was not opened before this call.

`SXTIOCSWTCH`
: Set the controlling channel. Possible errors include:

  `EINVAL`
  : An invalid channel number was given.

  `EPERM`
  : The command was not executed from channel 0.

`SXTIOCWF`
: Cause a channel to wait until it is the controlling channel. This command will return the error, `EINVAL`, if an invalid channel number is given.

`SXTIOCUBLK`
: Turn off the `loblk` control flag in the virtual tty of the indicated channel. The error `EINVAL` will be

returned if an invalid number or channel 0 is given.

`SXTIOCSTAT`
Get the status (blocked on input or output) of each channel and store in the `sxtblock` structure referenced by the argument. The error `EFAULT` will be returned if the structure cannot be written.

FILES

`/dev/sxt/??[0-7]`
Virtual tty devices

SEE ALSO

`shl(1)`, `stty(1)`, `ioctl(2)`, `open(2)`, `termio(7)`.

tc4(7) - zero(7)

tc4(7)
RM400 only

NAME

`tc4 -` four port serial interface

DESCRIPTION

There are currently two types of `tc4` cards available for Reliant UNIX, each card contains four serial interfaces. The first `tc4` supports four V24 interfaces, while the second `tc4` supports four daughter boards which permit the card to support other interface standards. Both cards are handled identically by Reliant UNIX and a user can see no logical difference between the cards.

Each Reliant UNIX system can normally support up to four `tc4` cards, each serial channel has an associated logical device which is accessed through special files in the `/dev/term` directory.

The `tc4` is currently controlled through the `sduart` driver. The `minor device numbers` of the `tc4` start at 80 and each card contains 4 logical devices, so that the first port on the first card has `minor number 80`, while the fourth port on the fourth card has `minor number 111`.

At the moment, all serial devices controlled by the `sduart` driver automatically push the modules `ldterm(7)` and `ttcompat(7)` (`/etc/ap/cons.ap`).

Each `tc4` card contains four switches which determine the physical I/O address of the card, no two cards can have the same physical I/O address. The first logical card, is the card with the lowest physical I/O address, the next logical card is the card with the next higher physical I/O address etc. There need be no relationship between which slot the card is inserted in and the physical I/O address of the card. Reliant UNIX only uses the physical I/O address to differentiate between different cards.

MODEM SIGNALS

Each `tc4` serial interface has three modem control lines. These are M1 M5 and M2, M3 is NOT supported on the card. By default the driver responds to modem signals. If a logical device "X" is created which has a `minor device number` of "X" + 256, then all modem signals are ignored. Which modem signals are supported on each line can also be controlled on a per serial interface basis by the editing of the

`/etc/conf/pack.d/tc4/space.c`

file. If this file is edited, the kernel must be relinked and the system rebooted before the changes can take effect.

The driver supports the usual `ioctl`s for a tty. Additionally, the driver handles the following `ioctl`s with the form:

```
ioctl (fd, I_STR, &arg)
int fd
struct strioctl arg
```

The structure `strioctl` is defined in `<stropts.h>` and has to be filled in the following form:

```
ic_cmd    = ioctl request (TIOCMODS, TIOCMODG, TIOCMGET)
ic_timout = 0
ic_len    = sizeof(int modem_flags)
ic_dp     = (char *)&modem_flags
```

`modem_flags`

`0x00000001`: Modem signal M1 (DSR)
`0x00000002`: Modem signal M2 (CTS)
`0x00000004`: Modem signal M5 (DCD)

`TIOCMODS`
: Sets the modem signals to be controlled by the driver (M1, M2, M3, M5) into `modem_flags`.

`TIOCMODG`
: Gets the current modem signals to be controlled by the driver. The driver sets the signals (flags) into `modem_flags`.

`TIOCMGET`
: The driver sets the current state of the modem signals into `modem_flags` and returns the state to the user.

FILES

`/dev/term/{80-111}`
`/dev/term/{336-367}`
`/etc/conf/pack.d/tc4/space.c`
`/etc/ap/cons.ap`

SEE ALSO

`ldterm(7)`, `termio(7)`, `ttcompat(7)`.
Programmer's Guide: STREAMS.

System Administrator's Reference Manual

tca(7)
RM600 only

NAME

`tca` - asynchronous terminal controller

DESCRIPTION

Asynchronous terminals or devices are operated with a speed of up to 38400 bits/sec. on the RM600 via the TCA controller. A TCA controller (board) is divided into 2 logical controllers each with 8 ports.

The terminals or devices are connected to the TCA controllers via connection boxes.

The boxes are connected to the TCA controllers via 2 64-core cables. The upper 64-pin plug on the TCA board (channel 0) provides the even logical controller numbers whereas the lower plug (channel 1) provides the uneven logical controller numbers.

Eight terminals or devices (SW8: I0-I7 or VB8: V0-V7) can be connected to each box via 9-pin plugs.

At present, a maximum of 16 TCA boards or 256 terminal ports are supported.

The system is configured using `Config` (see `sysadm(1M)` , menu option "Configuration"). Two serial numbers must be assigned to every TCA controller in `/etc/controller`. The "channel-0" number is 0 and the "channel-1" number is 1:

ctlr_type	:	*log_ctlr_no*	:	*bus_type*	:	*bus_id*	:	*slot_id*[	:	*channel*]
scsi	:	0	:	mb2	:	0	:	0		
lce	:	0	:	mb2	:	0	:	2		
tca	:	0	:	mb2	:	0	:	8	:	0
tca	:	1	:	mb2	:	0	:	8	:	1
tca	:	2	:	mb2	:	0	:	1	:	0
tca	:	3	:	mb2	:	0	:	1	:	1
sim	:	0	:	mb2	:	0	:	5		
sih	:	0	:	mb2	:	0	:	6		

The NVRAM must then be loaded with `/etc/ldcontroller` and the system restarted.

The terminal is configured with either `termadd(1M)` (see the following example) or with `Config`:

```
termadd –T 97801-5 –b tca0,7
termshow tc7
termdel  tc7
```

Two drivers (the SVA and TCA drivers) are used to operate the TCA controller. The SVA driver is used to load the TCA board software and to read service and statistical data from the TCA board. The loader is called when starting the system via the shell script `/etc/S61tca`.

The TCA driver is a STREAMS-based TTY driver.

Modem signal handling

When opening a TTY port [`open(2)`], the modem signals `S1/S2` are set. The message line `M5` is monitored [see `termio(7)`]. When closing the TTY port [`close(2)`], the `S1` and `S2` signals are reset again. No further characters are processed after `close` has been called.

Important: This behavior can result in data losses with printers if the printer sends CTRL-S/CTRL-Q (X-ON/X-OFF) between a `close` and an `open` call. To avoid this problem, see `tt_open(1M)` .

WARNINGS

The TCA controller does not support the speeds 50 and 134 bits/sec.

Hardware flow control and diagnostic tools are not available at present.

FILES

`/etc/init.d/tca.sh`
`/etc/S61tca`
`/etc/ap/tca.ap`
`/opt/etc/tca_sva_ppt/sva/sw_tca.omf`
`/sbin/mb2load`

SEE ALSO

`sysadm`(1M), `termadd`(1M), `tt_open`(1M), `controller`(4), `termio`(7), `ldcontroller`(8).

On `Config`: □

Manual *System Administration and Hardware Configuration Using the SYSADM User Interface*.

System Administrator's Reference Manual

termio(7)

NAME

termio - general terminal interface

SYNOPSIS

```
#include <termio.h>
ioctl(int fildes, int request, struct termio *arg);
ioctl(int fildes, int request, int arg);
#include <termios.h>
ioctl(int fildes, int request, struct termios *arg);
```

DESCRIPTION

Reliant UNIX supports a general interface for asynchronous communications ports that is hardware-independent. The user interface to this functionality is via function calls described in termios(3C) or ioctl commands described in this section. This section also discusses the common features of the terminal subsystem which are relevant with both user interfaces.

When a terminal file is opened, it normally causes the process to wait until a connection is established. In practice, users' programs seldom open terminal files; they are opened by the system and become a user's standard input, output, and error files. The very first terminal file opened by the session leader, which is not already associated with a session, becomes the controlling terminal for that session. The controlling terminal plays a special role in handling quit and interrupt signals, as discussed below. The controlling terminal is inherited by a child process during a fork(2). A process can break this association by changing its session using setsid(2).

A terminal associated with one of these files ordinarily operates in full-duplex mode. Characters may be typed at any time, even during output and are only lost when the character input buffers of the system become completely full, which is rare (e.g. if the number of characters in the line discipline buffer exceeds {MAX_CANON} and IMAXBEL [see below] is not set), or when the user has accumulated {MAX_INPUT} number of input characters that have not yet been read by some program. When the input limit is reached, all the characters saved in the buffer up to that point are thrown away without notice.

Session management (Job Control)

A control terminal will distinguish one of the process groups in the session associated with it to be the foreground process group. All other process groups in the session are designated as background process groups. This foreground process group plays a special role in handling signal-generating input characters, as discussed below. By default, when a controlling terminal is allocated, the controlling process's process group is assigned as foreground process group.

Background process groups in the controlling process's session are subject to a job control line discipline when they attempt to access their controlling terminal. Process groups can be sent signals that will cause them to stop, unless they have made other arrangements. An exception is made for members of orphaned process groups. These are process groups which do not have a member with a parent in another process group that is in the same session and therefore shares the same controlling terminal. When a member's orphaned process group attempts to access its controlling terminal, errors will be returned, since there is no process to continue it should it stop.

If a member of a background process group attempts to read its controlling terminal, its process group will be sent a SIGTTIN signal, which will normally cause the members of that process group to stop. If, however, the process is ignoring or holding SIGTTIN, or is a member of an orphaned process group, the read will fail with errno set to EIO, and no signal will be sent.

If a member of a background process group attempts to write its controlling terminal and the TOSTOP bit is set in the c_lflag field, its process group will be sent a SIGTTOU signal, which will normally cause the members

of that process group to stop. If, however, the process is ignoring or holding SIGTTOU, the write will succeed. If the process is not ignoring or holding SIGTTOU and is a member of an orphaned process group, the write will fail with errno set to EIO, and no signal will be sent.

If TOSTOP is set and a member of a background process group attempts to ioctl its controlling terminal, and that ioctl will modify terminal parameters (e.g. TCSETA, TCSETAW, TCSETAF, or TIOCSPGRP), its process group will be sent a SIGTTOU signal, which will normally cause the members of that process group to stop. If, however, the process is ignoring or holding SIGTTOU, the ioctl will succeed. If the process is not ignoring or holding SIGTTOU and is a member of an orphaned process group, the write will fail with errno set to EIO, and no signal will be sent.

Canonical mode input processing

Normally, terminal input is processed in units of lines. A line is delimited by a newline (ASCII LF) character, an end-of-file (ASCII EOT) character, or an end-of-line character. This means that a program attempting to read will be suspended until an entire line has been typed. Also, no matter how many characters are requested in the read call, at most one line will be returned. It is not necessary, however, to read a whole line at once; any number of characters may be requested in a read, even one, without losing information.

During input, erase and kill processing is normally done. The ERASE character (by default, the character number #) erases the last character typed. The WERASE character (the character CTRL-W) erases the last "word" typed in the current input line (but not any preceding spaces or tabs). A "word" is defined as a sequence of non-blank characters, with tabs counted as blanks. Neither ERASE nor WERASE will erase beyond the beginning of the line. The KILL character (by default, the character @) kills (deletes) the entire input line, and optionally outputs a newline character. All these characters operate on a key stroke basis, independent of any backspacing or tabbing that may have been done. The REPRINT character (the character CTRL-R) prints a newline followed by all characters that have not been read. Reprinting also occurs automatically if characters that would normally be erased from the screen are fouled by program output. The characters are reprinted as if they were being echoed; consequencely, if ECHO is not set, they are not printed.

The ERASE and KILL characters may be entered literally by preceding them with the escape character (\ ). In this case, the escape character is not read. The erase and kill characters may be changed.

Non-canonical mode input processing

In non-canonical mode input processing, input characters are not assembled into lines, and erase and kill processing does not occur. The MIN and TIME values are used to determine how to process the characters received.

MIN represents the minimum number of characters that should be received when the read is satisfied (i.e. when the characters are returned to the user). TIME is a timer of 0.10-second granularity that is used to timeout bursty and short-term data transmissions. The four possible values for MIN and TIME and their interactions are described below.

Case A: MIN > 0, TIME > 0

In this case, TIME serves as an intercharacter timer and is activated after the first character is received. Since it is an intercharacter timer, it is reset after a character is received. The interaction between MIN and TIME is as follows: as soon as one character is received, the intercharacter timer is started. If MIN characters are received before the intercharacter timer expires (note that the timer is reset upon receipt of each character), the read is satisfied. If the timer expires before MIN characters are received, the characters received to that point are returned to the user. Note that if TIME expires, at least one character will be returned because the timer would not have been enabled unless a character was received. In this case (MIN > 0, TIME > 0), the read sleeps until the MIN and TIME mechanisms are activated by the receipt of the first character. If the number of characters read is less than the number of characters available, the timer is not reactivated and the subsequent read is satisfied immediately.

Case B: MIN > 0, TIME = 0

In this case, since the value of TIME is zero, the timer plays no role and only MIN is significant. A pending read is not satisfied until MIN characters are received (the pending read sleeps until MIN

characters are received). A program that uses this case to read record based terminal I/O may block indefinitely in the read operation.

Case C: MIN = 0, TIME > 0
: In this case, since MIN = 0, TIME no longer represents an intercharacter timer: it now serves as a read timer that is activated as soon as a read is done. A read is satisfied as soon as a single character is received or the read timer expires. Note that, in this case, if the timer expires, no character is returned. If the timer does not expire, the only way the read can be satisfied is if a character is received. In this case, the read will not block indefinitely waiting for a character; if no character is received within TIME*.10 seconds after the read is initiated, the read returns with zero characters.

Case D: MIN = 0, TIME = 0
: In this case, return is immediate. The minimum of either the number of characters requested or the number of characters currently available is returned without waiting for more characters to be input.

Comparison of the different cases of MIN, TIME interaction

Some points to note about MIN and TIME:

1. In the following explanations, note that the interactions of MIN and TIME are not symmetric. For example, when MIN > 0 and TIME = 0, TIME has no effect. However, in the opposite case, where MIN = 0 and TIME > 0, both MIN and TIME play a role in that MIN is satisfied with the receipt of a single character.
2. Also note that in case A (MIN > 0, TIME > 0), TIME represents an intercharacter timer, whereas in case C (TIME = 0, TIME > 0), TIME represents a read timer.

These two points highlight the dual purpose of the MIN/TIME feature. Cases A and B, where MIN > 0, exist to handle burst mode activity (e.g. file transfer programs), where a program would like to process at least MIN characters at a time. In case A, the intercharacter timer is activated by a user as a safety measure; in case B, the timer is turned off.

Cases C and D exist to handle single character, timed transfers. These cases are readily adaptable to screen-based applications that need to know if a character is present in the input queue before refreshing the screen. In case C, the read is timed, whereas in case D, it is not.

Another important note is that MIN is always just a minimum. It does not denote a record length. For example, if a program does a read of 20 bytes, MIN is 10, and 25 characters are present, then 20 characters will be returned to the user.

Writing characters

When one or more characters are written, they are transmitted to the terminal as soon as previously written characters have finished typing. Input characters are echoed as they are typed if echoing has been enabled. If a process produces characters more rapidly than they can be typed, it will be suspended when its output queue exceeds some limit. When the queue is drained down to some threshold, the program is resumed.

Special characters

Certain characters have special functions on input. These functions and their default character values are summarized as follows:

INTR (Rubout or ASCII DEL) generates a SIGINT signal. SIGINT is sent to all frequent processes associated with the controlling terminal. Normally, each such process is forced to terminate, but arrangements may be made either to ignore the signal or to receive a trap to an agreed upon location [see signal(5)].

QUIT (CTRL-| or ASCII FS) generates a SIGQUIT signal. Its treatment is identical to the interrupt signal except that, unless a receiving process has made other arrangements, it will not only be terminated but a core image file (called core) will be created in the current working directory.

ERASE (#) erases the preceding character. It does not erase beyond the start of a line, as delimited by a NL, EOF, EOL, or EOL2 character.

WERASE

(CTRL-W or ASCII ETX) erases the preceding "word". It does not erase beyond the start of a line, as delimited by a NL, EOF, EOL, or EOL2 character.

KILL (@) deletes the entire line, as delimited by a NL, EOF, EOL, or EOL2 character.

REPRINT
(CTRL-R or ASCII DC2) reprints all characters, preceded by a newline, that have not been read.

EOF (CTRL-D or ASCII EOT) may be used to generate an end-of-file from a terminal. When received, all the characters waiting to be read are immediately passed to the program, without waiting for a newline, and the EOF is discarded. Thus, if no characters are waiting (i.e. the EOF occurred at the beginning of a line) zero characters are passed back, which is the standard end-of-file indication. Unless escaped, the EOF character is not echoed. Because EOT is the default EOF character, this prevents terminals that respond to EOT from hanging up.

NL (ASCII LF) is the normal line delimiter. It cannot be changed or escaped.

EOL (ASCII NULL) is an additional line delimiter, like NL. It is not normally used.

EOL2 is another additional line delimiter.

SWTCH (CTRL-Z or ASCII EM) is used only when `shl` layers is invoked.

SUSP (CTRL-Z or ASCII SUB) generates a `SIGTSTP` signal. `SIGTSTP` stops all processes in the foreground process group for that terminal.

DSUSP (CTRL-Y or ASCII EM) It generates a `SIGTSTP` signal as SUSP does, but the signal is sent when a process in the foreground process group attempts to read the DSUSP character, rather than when it is typed.

STOP (CTRL-S or ASCII DC3) can be used to suspend output temporarily. It is useful with CRT terminals to prevent output from disappearing before it can be read. While output is suspended, STOP characters are ignored and not read.

START (CTRL-Q or ASCII DC1) is used to resume output. Output has been suspended by a STOP character. While output is not suspended, START characters are ignored and not read.

DISCARD
(CTRL-O or ASCII SI) causes subsequent output to be discarded. Output is discarded until another DISCARD character is typed, more input arrives, or the condition is cleared by a program.

LNEXT (CTRL-V or ASCII SYN) causes the special meaning of the next character to be ignored. This works for all the special characters mentioned above. It allows characters to be input that would otherwise be interpreted by the system (e.g. KILL, QUIT).

The character values for INTR, QUIT, ERASE, WERASE, KILL, REPRINT, EOF, EOL, EOL2, SWTCH, SUSP, DSUSP, STOP, START, DISCARD, and LNEXT may be changed to suit individual tastes. If the value of a special control character is `_POSIX_VDISABLE` (0), the function of that special control character is disabled. The ERASE, KILL, and EOF characters may be escaped by a preceding `\` character, in which case no special function is done. Any of the special characters may be preceded by the LNEXT character, in which case no special function is done.

Modem disconnect

When a modem disconnect is detected, a `SIGHUP` signal is sent to the terminal's controlling process. Unless other arrangements have been made, these signals cause the process to terminate. If `SIGHUP` is ignored or caught, any subsequent read returns with an end-of-file indication until the terminal is closed.

If the controlling process is not in the foreground process group of the terminal, a `SIGTSTP` is sent to the terminal's foreground process group. Unless other arrangements have been made, these signals cause the processes to stop.

Processes in background process groups that attempt to access the controlling terminal after modem disconnect while the terminal is still allocated to the session will receive appropriate `SIGTTOU` and `SIGTTIN` signals. Unless other arrangements have been made, this signal causes the processes to stop.

The controlling terminal will remain in this state until it is reinitialized with a successful open by the controlling

process, or deallocated by the controlling process.

Terminal parameters

The parameters that control the behavior of devices and modules providing the `termios` interface are specified by the `termios` structure defined by `<termios.h>`. Several `ioctl(2)` system calls that fetch or change these parameters use this structure that contains the following members:

```
tcflag_t c_iflag;     /* input modes */
tcflag_t c_oflag;     /* output modes */
tcflag_t c_cflag;     /* control modes */
tcflag_t c_lflag;     /* local modes */
cc_t     c_cc[NCCS];  /* control chars */
```

The special control characters are defined by the array `c_cc`. The symbolic name `NCCS` is the size of the control-character array and is also defined by `<termios.h>`. The relative positions, subscript names, and typical default values for each function are as follows:

0	`VINTR`	DEL
1	`VQUIT`	FS
2	`VERSE`	#
3	`VKILL`	@
4	`VEOF`	EOT
5	`VEOL`	NUL
6	`VEOL2`	NUL
7	`VSWTCH`	NUL
8	`VSTRT`	DC1
9	`VSTOP`	DC3
10	`VSUSP`	SUB
11	`VDSUSP`	EM
12	`VREPRINT`	DC2
13	`VDISCRD`	SI
14	`VWERSE`	ETB
15	`VLNEXT`	SYN
16-19	`reserved`	

Input modes

The `c_iflag` field describes the basic terminal input control:

`IGNBRK` Ignore break condition.

`BRKINT` Signal interrupt on break.

`IGNPAR` Ignore characters with parity errors.

`PARMRK` Mark parity errors.

`INPCK` Enable input parity check.

`ISTRIP` Strip character.

`INLCR` Map NL to CR on input.

IGNCR Ignore CR.

ICRNL Map CR to NL on input.

IUCLC Map uppercase to lowercase on input.

IXON Enable start/stop output control.

IXANY Enable any character to restart output.

IXOFF Enable start/stop input control.

IMAXBEL Echo BEL on input line too long.

If IGNBRK is set, a break condition (a character framing error with data all zeros) detected on input is ignored, that is, not put on the input queue and therefore not read by any process. If IGNBRK is not set and BRKINT is set, the break condition shall flush the input and output queues and if the terminal is the controlling terminal of a foreground process group, the break condition generates a single SIGINT signal to that foreground process group. If neither IGNBRK nor BRKINT is set, a break condition is read as a single ASCII NULL character ('\0'), or if PARMRK is set, as '\377', '\0', '\0'.

If IGNPAR is set, a byte with framing or parity errors (other than break) is ignored.

If PARMRK is set, and IGNPAR is not set, a byte with a framing or parity error (other than break) is given to the application as the three-character sequence: '\377', '\0', X, where X is the data of the byte received in error. To avoid ambiguity in this case, if ISTRIP is not set, a valid character of '\377' is given to the application as '\377', '\377'. If neither IGNPAR nor PARMRK is set, a framing or parity error (other than break) is given to the application as a single ASCII NULL character ('\0').

If INPCK is set, input parity checking is enabled. If INPCK is not set, input parity checking is disabled. This allows output parity generation without input parity errors. Note that whether input parity checking is enabled or disabled is independent of whether parity detection is enabled or disabled. If parity detection is enabled but input parity checking is disabled, the hardware to which the terminal is connected will recognize the parity bit, but the terminal special file will not check whether this is set correctly or not.

If ISTRIP is set, valid input characters are first stripped to seven bits, otherwise all eight bits are processed.

If INLCR is set, a received NL character is translated into a CR character. If IGNCR is set, a received CR character is ignored (not read). Otherwise, if ICRNL is set, a received CR character is translated into a NL character.

If IUCLC is set, a received uppercase, alphabetic character is translated into the corresponding lowercase character.

If IXON is set, start/stop output control is enabled. A received STOP character suspends output and a received START character restarts output. The STOP and START characters will not be read, but will merely perform flow control functions. If IXANY is set, any input character restarts output that has been suspended.

If IXOFF is set, the system transmits a STOP character when the input queue is nearly full, and a START character when enough input has been read so that the input queue is nearly empty again.

If IMAXBEL is set, the ASCII BEL character is echoed if the input stream overflows. Further input is not stored, but any input already present in the input stream is not disturbed. If IMAXBEL is not set, no BEL character is echoed, and all input present in the input queue is discarded if the input stream overflows.

The initial input control value is BRKINT, ICRNL, IXON, ISTRIP.

Output modes

The c_oflag field specifies the system treatment of output:

OPOST Post-process output.

OLCUC Map lowercase to upper on output.

ONLCR Map NL to CR-NL on output.

OCRNL Map CR to NL on output.

ONOCR No CR output at column 0.

ONLRET NL performs CR function.

OFILL Use fill characters for delay.

OFDEL Fill is DEL, else NULL.

NLDLY Select newline delays:

NL0

NL1

CRDLY Select carriage-return delays:

CR0

CR1

CR2

CR3

TABDLY Select horizontal tab delays:

TAB0 or tab expansion:

TAB1

TAB2

TAB3 Expand tabs to spaces.

XTABS Expand tabs to spaces.

BSDLY Select backspace delays:

BS0

BS1

VTDLY Select vertical tab delays:

VT0

VT1

FFDLY Select form feed delays:

FF0

FF1

If OPOST is set, output characters are post-processed as indicated by the remaining flags; otherwise, characters are transmitted without change.

If OLCUC is set, a lowercase alphabetic character is transmitted as the corresponding uppercase character. This function is often used in conjunction with IUCLC.

If ONLCR is set, the NL character is transmitted as the CR-NL character pair. If OCRNL is set, the CR character is transmitted as the NL character. If ONOCR is set, no CR character is transmitted when at column 0 (first position). If ONRET is set, the NL character is assumed to do the carriage-return function; the column pointer is set to 0 and the delays specified for CR are used. Otherwise, the NL character is assumed to do just the line-feed function; the column pointer remains unchanged. The column pointer is also set to 0 if the CR character is actually transmitted.

The delay bits specify how long transmission stops to allow for mechanical or other movement when certain characters are sent to the terminal. In all cases, a value of 0 indicates no delay. If OFILL is set, fill characters are transmitted for delay instead of a timed delay. This is useful for high baud rate terminals that need only a minimal delay. If OFDEL is set, the fill character is DEL; otherwise it is NULL.

If a form-feed or vertical-tab delay is specified, it lasts for about 2 seconds.

Newline delay lasts about 0.10 seconds. If `ONLRET` is set, the carriage-return delays are used instead of the newline delays. If `OFILL` is set, two fill characters are transmitted.

Carriage-return delay type 1 is dependent on the current column position, type 2 is about 0.10 seconds, and type 3 is about 0.15 seconds. If `OFILL` is set, delay type 1 transmits two fill characters, and type 2 transmits four fill characters.

Horizontal-tab delay type 1 is dependent on the current column position. Type 2 is about 0.10 seconds. Type 3 specifies that tabs are to be expanded into spaces. If `OFILL` is set, two fill characters are transmitted for any delay.

Backspace delay lasts about 0.05 seconds. If `OFILL` is set, one fill character is transmitted.

The actual delays depend on line speed and system load.

The initial output control value is `OPOST`, `ONLCR`, `TAB3`.

Control Modes

The `c_cflag` field describes the hardware control of the terminal:

`CBAUD`	Baud rate:
`B0`	Hang up
`B50`	50 baud
`B75`	75 baud
`B110`	110 baud
`B134`	134 baud
`B150`	150 baud
`B200`	200 baud
`B300`	300 baud
`B600`	600 baud
`B1200`	1200 baud
`B1800`	1800 baud
`B2400`	2400 baud
`B4800`	4800 baud
`B9600`	9600 baud
`B19200`	19200 baud
`EXTA`	External A
`B38400`	38400 baud
`EXTB`	External B
`CSIZE`	Character size:
`CS5`	5 bits
`CS6`	6 bits
`CS7`	7 bits
`CS8`	8 bits
`CSTOPB`	Send two stop bits, else one
`CREAD`	Enable receiver

PARENB Parity enable

PARODD Odd parity, else even

HUPCL Hang up on last close

CLOCAL Local line, else dial-up

CIBAUD Input baud rate, if different from output rate

PAREXT Extended parity for mark and space parity

The CBAUD bits specify the baud rate. The zero baud rate, B0, is used to hang up the connection. If B0 is specified, the data-terminal-ready signal is not asserted. Normally, this disconnects the line. If the CIBAUD bits are not zero, they specify the input baud rate, with the CBAUD bits specifying the output baud rate; otherwise, the output and input baud rates are both specified by the CBAUD bits. The values for the CIBAUD bits are the same as the values for the CBAUD bits, shifted left IBSHIFT bits. For any particular hardware, impossible speed changes are ignored.

The CSIZE bits specify the character size in bits for both transmission and reception. This size does not include the parity bit, if any. If CSTOPB is set, two stop bits are used; otherwise, one stop bit is used. For example, at 110 baud, two stops bits are required.

If PARENB is set, parity generation and detection is enabled, and a parity bit is added to each character. If parity is enabled, the PARODD flag specifies odd parity if set; otherwise, even parity is used.

If CREAD is set, the receiver is enabled. Otherwise, no characters are received.

If HUPCL is set, the line is disconnected when the last process with the line open closes it or terminates. That is, the data-terminal-ready signal is not asserted.

If CLOCAL is set, the line is assumed to be a local, direct connection with no modem control; otherwise, modem control is assumed.

The initial hardware control value after open is B300, CS8, CREAD, HUPCL.

Local modes

The c_lflag field of the argument structure is used by the line discipline to control terminal functions. The basic line discipline provides the following:

ISIG Enable signals.

ICANON Canonical input (erase and kill processing).

XCASE Canonical upper/lower presentation.

ECHO Enable echo.

ECHOE Echo erase character as BS-SP-BS.

ECHOK Echo NL after kill character.

ECHONL Echo NL.

NOFLSH Disable flush after interrupt or quit.

TOSTOP Send SIGTTOU for background output.

ECHOCTL Echo control characters as ^*char*, delete as ^?.

ECHOPRT Echo erase character as character erased.

ECHOKE BS-SP-BS erase entire line on line kill.

FLUSHO Output is being flushed.

PENDIN Retype pending input at next read or input character.

IEXTEN Enable extended (implementation-defined) functions.

If ISIG is set, each input character is checked against the special control characters INTR, QUIT, SWTCH, SUSP, STATUS, and DSUSP. If an input character matches one of these control characters, the function associated with that character is performed. If ISIG is not set, no checking is done. Thus, these special input functions are possible only if ISIG is set.

If ICANON is set, canonical processing is enabled. This enables the erase and kill edit functions, and the assembly of input characters into lines delimited by NL, EOF, EOL, and EOL2. If ICANON is not set, read requests are satisfied directly from the input queue. A read is not satisfied until at least MIN characters have been received or the timeout value TIME has expired between characters. This allows fast bursts of input to be read efficiently while still allowing single character input. The time value represents tenths of seconds.

If XCASE is set, and if ICANON is set, an uppercase letter is accepted on input by preceding it with a \ character, and is output preceded by a \ character. In this mode, the following escape sequences are generated on output and accepted on input:

for:	use:
`	\'
|	\!
~	\^
{	\(
}	\)
\	\\

For example, A is input as \a, \n as \\n, and \N as \\\n.

If ECHO is set, characters are echoed as received.

When ICANON is set, the following echo functions are possible.

1. If ECHO and ECHOE are set, and ECHOPRT is not set, the ERASE and WERASE characters are echoed as one or more ASCII BS SP BS, which clears the last character(s) from a CRT screen.
2. If ECHO and ECHOPRT are set, the first ERASE and WERASE character in a sequence echoes as a backslash (\), followed by the characters being erased. Subsequent ERASE and WERASE characters echo the characters being erased, in reverse order. The next non-erase character causes a slash (/) to be typed before it is echoed. ECHOPRT should be used for hard copy terminals.
3. If ECHOKE is set, the kill character is echoed by erasing each character on the line from the screen (using the mechanism selected by ECHOE and ECHOPRT).
4. If ECHOK is set, and ECHOKE is not set, the NL character is echoed after the kill character to emphasize that the line is deleted. Note that an escape character (\ ) or an LNEXT character preceding the erase or kill character removes any special function.
5. If ECHONL is set, the NL character is echoed even if ECHO is not set. This is useful for terminals set to local echo (so called half-duplex).

If ECHOCTL is set, all control characters (characters with codes between 0 and 37 octal) other than ASCII TAB, ASCII NL, the START character, and the STOP character, ASCII CR, and ASCII BS are echoed as ^*X*, where *X* is the character given by adding 100 octal to the code of the control character (so that the character with octal code 1 is echoed as ^A), and the ASCII DEL character, with code 177 octal, is echoed as ^?.

If NOFLSH is set, the normal flush of the input and output queues associated with the INTR, QUIT, and SUSP characters is not done. This bit should be set when restarting system calls that read from or write to a terminal [see sigaction(2)].

If TOSTOP is set, the signal SIGTTOU is sent to a process that tries to write to its controlling terminal if it is not in the foreground process group for that terminal. This signal normally stops the process. Otherwise, the output generated by that process is output to the current output stream. Processes that are blocking or ignoring SIGTTOU signals are excepted and allowed to produce output, if any.

If FLUSHO is set, data written to the terminal is discarded. This bit is set when the FLUSH character is typed. A program can cancel the effect of typing the FLUSH character by clearing FLUSHO.

If PENDIN is set, any input that has not yet been read is reprinted when the next character arrives as input.

If IEXTEN is set, the following implementation-defined functions are enabled: special characters (WERASE, REPRINT, DISCARD, and LNEXT) and local flags (TOSTOP, ECHOCTL, ECHOPRT, ECHOKE, FLUSHO, and PENDIN).

The initial line-discipline control value is ISIG, ICANON, ECHO, ECHOK.

Minimum and Timeout

The MIN and TIME values are described above under **Non-canonical mode input processing**. The initial value of MIN is 1, and the initial value of TIME is 0.

Terminal size

The number of lines and columns on the terminal's display is specified in the winsize structure defined by <sys/termios.h> and includes the following members:

```
unsigned short ws_row;     /* rows, in characters */
unsigned short ws_col;     /* columns, in characters */
unsigned short ws_xpixel;  /* horizontal size, in pixels */
unsigned short ws_ypixel;  /* vertical size, in pixels */
```

termio structure

The Reliant UNIX termio structure is used by some ioctls; it is defined by <sys/termio.h> and includes the following members:

```
unsigned short c_iflag;    /* input modes */
unsigned short c_oflag;    /* output modes */
unsigned short c_cflag;    /* control modes */
unsigned short c_lflag;    /* local modes */
char           c_line;     /* line discipline */
unsigned char  c_cc[NCC];  /* control chars */
```

The special control characters are defined by the array c_cc. The symbolic name NCC is the size of the control-character array and is also defined by <termio.h>. The relative positions, subscript names, and typical default values for each function are as follows:

0	VINTR	DEL
1	VQUIT	FS
2	VERASE	#
3	VKILL	@
4	VEOF	EOT
5	VEOL	NUL
6	VEOL2	NUL
7	reserved	

The calls that use the termio structure only affect the flags and control characters that can be stored in the termio structure; all other flags and control characters are unaffected.

Modem lines

On special files representing serial ports, the modem control lines supported by the hardware can be read, and the modem status lines supported by the hardware can be changed. The following modem control and status lines may be supported by a device; they are defined by <sys/termios.h>:

TIOCM_LE	line enable
TIOCM_DTR	data terminal ready

TIOCM_RTS	request to send
TIOCM_ST	secondary transmit
TIOCM_SR	secondary receive
TIOCM_CTS	clear to send
TIOCM_CAR	carrier detect
TIOCM_RNG	ring
TIOCM_DSR	data set ready

TIOCM_CD is a synonym for TIOCM_CAR, and TIOCM_RI is a synonym for TIOCM_RNG. Not all of these are necessarily supported by any particular device; check the manual page for the device in question.

IOCTLS

The ioctls supported by devices and STREAMS modules providing the termios interface are listed below. Some calls may not be supported by all devices or modules. The functionality provided by these calls is also available through the preferred function call interface specified on termios(3C).

TCGETS
: The argument is a pointer to a termios structure. The current terminal parameters are fetched and stored into that structure.

TCSETS
: The argument is a pointer to a termios structure. The current terminal parameters are set from the values stored in that structure. The change is immediate.

TCSETSW
: The argument is a pointer to a termios structure. The current terminal parameters are set from the values stored in that structure. The change occurs after all characters queued for output have been transmitted. This form should be used when changing parameters that affect output.

TCSETSF
: The argument is a pointer to a termios structure. The current terminal parameters are set from the values stored in that structure. The change occurs after all characters queued for output have been transmitted; all characters queued for input are discarded and then the change occurs.

TCGETA
: The argument is a pointer to a termio structure. The current terminal parameters are fetched, and those parameters that can be stored in a termio structure are stored into that structure.

TCSETA
: The argument is a pointer to a termio structure. Those terminal parameters that can be stored in a termio structure are set from the values stored in that structure. The change is immediate.

TCSETAW
: The argument is a pointer to a termio structure. Those terminal parameters that can be stored in a termio structure are set from the values stored in that structure. The change occurs after all characters queued for output have been transmitted. This form should be used when changing parameters that affect output.

TCSETAF
: The argument is a pointer to a termio structure. Those terminal parameters that can be stored in a termio structure are set from the values stored in that structure. The change occurs after all characters queued for output have been transmitted; all characters queued for input are discarded and then the change occurs.

TCSBRK
: The argument is an int value. Wait for the output to drain. If the argument is 0, then send a break (zero valued bits for 0.25 seconds).

TCXONC

Start/stop control. The argument is an `int` value. If the argument is 0, suspend output; if 1, restart suspended output; if 2, suspend input; if 3, restart suspended input.

`TCFLSH`
The argument is an `int` value. If the argument is 0, flush the input queue; if 1, flush the output queue; if 2, flush both the input and output queues.

`TIOCGPGRP`
The argument is a pointer to a `pid_t`. Set the value of that `pid_t` to the process group ID of the foreground process group associated with the terminal. See `termios(3C)` for a description or `TCGETPGRP`.

`TIOCSPGRP`
The argument is a pointer to a `pid_t`. Associate the process group whose process group ID is specified by the value of that `pid_t` with the terminal. The new process group value must be in the range of valid process group ID values. Otherwise, the error `EPERM` is returned. See `termios(3C)` for a description of `TCSETPGRP`.

`TIOCGSID`
The argument is a pointer to a `pid_t`. The session ID of the terminal is fetched and stored in the `pid_t`.

`TIOCGWINSZ`
The argument is a pointer to a `winsize` structure. The terminal driver's notion of the terminal size is stored into that structure.

`TIOCSWINSZ`
The argument is a pointer to a `winsize` structure. The terminal driver's notion of the terminal size is set from the values specified in that structure. If the new sizes are different from the old sizes, a `SIGWINCH` signal is set to the process group of the terminal.

`TIOCMBIS`
The argument is a pointer to an `int` whose value is a mask containing modem control lines to be turned on. The control lines whose bits are set in the argument are turned on; no other control lines are affected.

`TIOCMBIC`
The argument is a pointer to an `int` whose value is a mask containing modem control lines to be turned off. The control lines whose bits are set in the argument are turned off; no other control lines are affected.

`TIOCMGET`
The argument is a pointer to an `int`. The current state of the modem status lines is fetched and stored in the `int` pointed to by the argument.

`TIOCMSET`
The argument is a pointer to an `int` containing a new set of modem control lines. The modem control lines are turned on or off, depending on whether the bit for that mode is set or clear.

Default Port-Parameter
The Default Port-Parameters are `HUPCL`, 9600 Baud, `CREAD`, `CS08`, Noparity.

The following `ioctl`s are supported by the common STREAMS based terminal driver (i.e. only for devices connected to ITP ports) to aid in the implementation of the trusted login path feature of the optional Commercial Security Product (CSP). The `ioctl`s will fail unless CSP is installed in the operating system.

`TIOCSSECURE`
The argument is ignored. Modem control for the line is always assumed (i.e. `CLOCAL` settings are ignored). Once turned on, this setting cannot be turned off. The setting is turned off automatically by the driver when the device is no longer open by any process.

`TIOCGSECURE`
The argument is ignored. The `ioctl` call returns 1 if the device has the `TIOCSSECURE` setting turned on so that modem control is always assumed otherwise 0 is returned.

FILES

Files in or under `/dev`

SEE ALSO

`fork`(2), `ioctl`(2), `setsid`(2), `signal`(2), `termios`(3C), `streamio`(7).

System Administrator's Reference Manual

termiox(7)

NAME

termiox - extended general terminal interface

DESCRIPTION

The extended general terminal interface supplements the termio(7) general terminal interface by adding support for asynchronous hardware flow control, isochronous flow control and clock modes, and local implementations of additional asynchronous features. Some systems may not support all of these capabilities because of either hardware or software limitations. Other systems may not permit certain functions to be disabled. In these cases the appropriate bits will be ignored. See <termiox.h> for your system to find out which capabilities are supported.

Hardware Flow Control Modes

Hardware flow control supplements the termio(7) IXON, IXOFF, and IXANY character flow control. Character flow control occurs when one device controls the data transfer of another device by the insertion of control characters in the data stream between devices. Hardware flow control occurs when one device controls the data transfer of another device using electrical control signals on wires (circuits) of the asynchronous interface. Isochronous hardware flow control occurs when one device controls the data transfer of another device by asserting or removing the transmit clock signals of that device. Character flow control and hardware flow control may be simultaneously set.

In asynchronous, full duplex applications, the use of the Electronic Industries Association's EIA-232-D Request To Send (RTS) and Clear To Send (CTS) circuits is the preferred method of hardware flow control. An interface to other hardware flow control methods is included to provide a standard interface to these existing methods.

The EIA-232-D standard specified only uni-directional hardware flow control – the Data Circuit-terminating Equipment or Data Communications Equipment (DCE) indicates to the Data Terminal Equipment (DTE) to stop transmitting data. The termiox interface allows both uni-directional and bi-directional hardware flow control; when bi-directional flow control is enabled, either the DCE or DTE can indicate to each other to stop transmitting data across the interface. Note: It is assumed that the asynchronous port is configured as a DTE. If the connected device is also a DTE and not a DCE, then DTE to DTE (e.g. terminal or printer connected to computer) hardware flow control is possible by using a null modem to interconnect the appropriate data and control circuits.

Clock Modes

Isochronous communication is a variation of asynchronous communication whereby two communicating devices may provide transmit and/or receive clock to each other. Incoming clock signals can be taken from the baud rate generator on the local isochronous port controller, from CCITT V.24 circuit 114, Transmitter Signal Element Timing – DCE source (EIA-232-D pin 15), or from CCITT V.24 circuit 115, Receiver Signal Element Timing – DCE source (EIA-232-D pin 17). Outgoing clock signals can be sent on CCITT V.24 circuit 113, Transmitter Signal Element Timing – DTE source (EIA-232-D pin 24), on CCITT V.24 circuit 128, Receiver Signal Element Timing – DTE source (no EIA-232-D pin), or not sent at all.

In terms of clock modes, traditional asynchronous communication is implemented simply by using the local baud rate generator as the incoming transmit and receive clock source and not outputting any clock signals.

Terminal Parameters

The parameters that control the behavior of devices providing the termiox interface are specified by the termiox structure, defined in the <sys/termiox.h> header file. Several ioctl(2) system calls that fetch or change these parameters use this structure:

```
#define NFF 5
struct termiox
```

```
{
    unsigned short x_hflag;       /* hardware flow control modes */
    unsigned short x_cflag;       /* clock modes */
    unsigned short x_rflag[NFF];  /* reserved modes */
    unsigned short x_sflag;       /* spare local modes */
};
```

The `x_hflag` field describes hardware flow control modes:

`RTSXOFF`	0000001	Enable RTS hardware flow control on input.
`CTSXON`	0000002	Enable CTS hardware flow control on output.
`DTRXOFF`	0000004	Enable DTR hardware flow control on input.
`CDXON`	0000010	Enable CD hardware flow control on output.
`ISXOFF`	0000020	Enable isochronous hardware flow control on input.

The EIA-232-D DTR and CD circuits are used to establish a connection between two systems. The RTS circuit is also used to establish a connection with a modem. Thus, both DTR and RTS are activated when an asynchronous port is opened. If DTR is used for hardware flow control, then RTS must be used for connectivity. If CD is used for hardware flow control, then CTS must be used for connectivity. Thus, RTS and DTR (or CTS and CD) cannot both be used for hardware flow control at the same time. Other mutual exclusions may apply, such as the simultaneous setting of the `termio(7)` `HUPCL` and the `termiox` `DTRXOFF` bits, which use the DTE ready line for different functions.

Variations of different hardware flow control methods may be selected by setting the appropriate bits. For example, bi-directional RTS/CTS flow control is selected by setting both the `RTSXOFF` and `CTSXON` bits and bi-directional DTR/CTS flow control is selected by setting both the `DTRXOFF` and `CTSXON`. Modem control or uni-directional CTS hardware flow control is selected by setting only the `CTSXON` bit.

As previously mentioned, it is assumed that the local asynchronous port (e.g. computer) is configured as a DTE. If the connected device (e.g. printer) is also a DTE, it is assumed that the device is connected to the computer's asynchronous port via a null modem that swaps control circuits (typically RTS and CTS). The connected DTE drives RTS and the null modem swaps RTS and CTS so that the remote RTS is received as CTS by the local DTE. In the case that `CTSXON` is set for hardware flow control, printer's lowering of its RTS would cause CTS seen by the computer to be lowered. Output to the printer is suspended until the printer's raising of its RTS, which would cause CTS seen by the computer to be raised.

If `RTSXOFF` is set, the Request To Send (RTS) circuit (line) will be raised, and if the asynchronous port needs to have its input stopped, it will lower the Request To Send (RTS) line. If the RTS line is lowered, it is assumed that the connected device will stop its output until RTS is raised.

If `CTSXON` is set, output will occur only if the Clear To Send (CTS) circuit (line) is raised by the connected device. If the CTS line is lowered by the connected device, output is suspended until CTS is raised.

If `DTRXOFF` is set, the DTE Ready (DTR) circuit (line) will be raised, and if the asynchronous port needs to have its input stopped, it will lower the DTE Ready (DTR) line. If the DTR line is lowered, it is assumed that the connected device will stop its output until DTR is raised.

If `CDXON` is set, output will occur only if the Received Line Signal Detector (CD) circuit (line) is raised by the connected device. If the CD line is lowered by the connected device, output is suspended until CD is raised.

If `ISXOFF` is set, and if the isochronous port needs to have its input stopped, it will stop the outgoing clock signal. It is assumed that the connected device is using this clock signal to create its output. Transit and receive clock sources are programmed using the `x_cflag` fields. If the port is not programmed for external clock generation, `ISXOFF` is ignored. Output isochronous flow control is supported by appropriate clock source programming using the `x_cflag` field and enabled at the remote connected device.

The `x_cflag` field specifies the system treatment of clock modes.

`XMTCLK`	0000007	Transmit clock source:

XCIBRG	0000000	Get transmit clock from internal baud rate generator.
XCTSET	0000001	Get transmit clock from transmitter signal element timing (DCE source) lead, CCITT V.24 circuit 114, EIA-232-D pin 15.
XCRSET	0000002	Get transmit clock from receiver signal element timing (DCE source) lead, CCITT V.24 circuit 115, EIA-232-D pin 17.
RCVCLK	0000070	Receive clock source.
RCIBRG	0000000	Get receive clock from internal baud rate generator.
RCTSET	0000010	Get receive clock from transmitter signal element timing (DCE source) lead, CCITT V.24 circuit 114, EIA-232-D pin 15.
RCRSET	0000020	Get receive clock from receiver signal element timing (DCE source) lead, CCITT V.24 circuit 115, EIA-232-D pin 17.
TSETCLK	0000700	Transmitter signal element timing (DTE source) lead, CCITT V.24 circuit 113, EIA-232-D pin 24, clock source:
TSETCOFF	0000000	TSET clock not provided.
TSETCRBRG	0000100	Output receive baud rate generator on circuit 113.
TSETCTBRG	0000200	Output transmit baud rate generator on circuit 113.
TSETCTSET	0000300	Output transmitter signal element timing (DCE source) on circuit 113.
TSETCRSET	0000400	Output receiver signal element timing (DCE source) on circuit 113.
RSETCLK	0007000	Receiver signal element timing (DTE source) lead, CCITT V.24 circuit 128, no EIA-232-D pin, clock source.
RSETCOFF	0000000	RSET clock not provided.
RSETCRBRG	0001000	Output receive baud rate generator on circuit 128.
RSETCTBRG	0002000	Output transmit baud rate generator on circuit 128.
RSETCTSET	0003000	Output transmitter signal element timing (DCE source) on circuit 128.
RSETCRSET	0004000	Output receiver signal element timing (DCE) on circuit 128.

If the XMTCLK field has a value of XCIBRG the transmit clock is taken from the hardware internal baud rate generator, as in normal asynchronous transmission. If XMTCLK = XCTSET the transmit clock is taken from the Transmitter Signal Element Timing (DCE source) circuit. If XMTCLK = XCRSET the transmit clock is taken from the Receiver Signal Element Timing (DCE source) circuit.

If the RCVCLK field has a value of RCIBRG the receive clock is taken from the hardware Internal Baud Rate Generator, as in normal asynchronous transmission. If RCVCLK = RCTSET the receive clock is taken from the Transmitter Signal Element Timing (DCE source) circuit. If RCVCLK = RCRSET the receive clock is taken from the Receiver Signal Element Timing (DCE source) circuit.

If the TSETCLK field has a value of TSETCOFF the Transmitter Signal Element Timing (DTE source) circuit is not driven. If TSETCLK = TSETCRBRG the Transmitter Signal Element Timing (DTE source) circuit is driven by the Receive Baud Rate Generator. If TSETCLK = TSETCTBRG the Transmitter Signal Element Timing (DTE source) circuit is driven by the Transmit Baud Rate Generator. If TSETCLK = TSETCTSET the Transmitter Signal Element Timing (DTE source) circuit is driven by the Transmitter Signal Element Timing (DCE source). If TSETCLK = TSETCRBRG the Transmitter Signal Element Timing (DTE source) circuit is driven by the Receiver Signal Element Timing (DCE source).

If the RSETCLK field has a value of RSETCOFF the Receiver Signal Element Timing (DTE source) circuit is not driven. If RSETCLK = RSETCRBRG the Receiver Signal Element Timing (DTE source) circuit is driven by the Receive Baud Rate Generator. If RSETCLK = RSETCTBRG the Receiver Signal Element Timing (DTE source) circuit is driven by the Transmit Baud Rate Generator. If RSETCLK = RSETCTSET the Receiver Signal Element

Timing (DTE source) circuit is driven by the Transmitter Signal Element Timing (DCE source). If `RSETCLK` = `RSETCRBRG` the Receiver Signal Element Timing (DTE source) circuit is driven by the Receiver Signal Element Timing (DCE source).

The `x_rflag` is reserved for future interface definitions and should not be used by any implementations. The `x_sflag` may be used by local implementations wishing to customize their terminal interface using the `termiox` `ioctl` system calls.

IOCTLS

The `ioctl`(2) system calls have the form:

`ioctl(`*fildes*`,` *command*`,` *arg*`)`
`struct termiox *`*arg*`;`

The commands using this form are:

`TCGETX` The argument is a pointer to a `termiox` structure. The current terminal parameters are fetched and stored into that structure.

`TCSETX`
: The argument is a pointer to a `termiox` structure. The current terminal parameters are set from the values stored in that structure. The change is immediate.

`TCSETXW`
: The argument is a pointer to a `termiox` structure. The current terminal parameters are set from the values stored in that structure. The change occurs after all characters queued for output have been transmitted. This form should be used when changing parameters that will affect output.

`TCSETXF`
: The argument is a pointer to a `termiox` structure. The current terminal parameters are set from the values stored in that structure. The change occurs after all characters queued for output have been transmitted; all characters queued for input are discarded and then the change occurs.

FILES

`/dev/*`

SEE ALSO

`stty`(1), `ioctl`(2), `termio`(7).

System Administrator's Reference Manual

timod(7)

NAME

`timod` - Transport Interface cooperating STREAMS module

DESCRIPTION

`timod` is a STREAMS module for use with the Transport Interface (TI) functions of the Network Services library. The `timod` module converts a set of `ioctl(2)` calls into STREAMS messages that may be consumed by a transport protocol provider which supports the Transport Interface. This allows a user to initiate certain TI functions as atomic operations.

The `timod` module must only be pushed onto a stream terminated by a transport protocol provider which supports the TI.

All STREAMS messages, with the exception of the message types generated from the `ioctl` commands described below, will be transparently passed to the neighboring STREAMS module or driver. The messages generated from the following `ioctl` commands are recognized and processed by the `timod` module. The format of the `ioctl` call is:

```
#include <sys/stropts.h>
  -
  -
struct strioctl strioctl;
  -
  -
strioctl.ic_cmd = cmd;
strioctl.ic_timeout = INFTIM;
strioctl.ic_len = size;
strioctl.ic_dp = (char *)buf
ioctl(fildes, I_STR, &strioctl);
```

Whereas when issued, *size* is the size of the appropriate TI message to be sent to the transport provider, on return *size* is the size of the appropriate TI message from the transport provider in response to the issued TI message. *buf* is a pointer to a buffer large enough to hold the contents of the appropriate TI messages. The TI message types are defined in `<sys/tihdr.h>`. The possible values for the *cmd* field are:

`TI_BIND`
: Bind an address to the underlying transport protocol provider. The message issued to the `TI_BIND` `ioctl` is equivalent to the TI message type `T_BIND_REQ` and the message returned by the successful completion of the `ioctl` is equivalent to the TI message type `T_BIND_ACK`.

`TI_UNBIND`
: Unbind an address from the underlying transport protocol provider. The message issued to the `TI_UNBIND` `ioctl` is equivalent to the TI message type `T_UNBIND_REQ` and the message returned by the successful completion of the `ioctl` is equivalent to the TI message type `T_OK_ACK`.

`TI_GETINFO`
: Get the TI protocol specific information from the transport protocol provider. The message issued to the `TI_GETINFO` `ioctl` is equivalent to the TI message type `T_INFO_REQ` and the message returned by the successful completion of the `ioctl` is equivalent to the TI message type `T_INFO_ACK`.

`TI_OPTMGMT`
: Get, set or negotiate protocol specific options with the transport protocol provider. The message issued to the `TI_OPTMGMT` `ioctl` is equivalent to the TI message type `T_OPTMGMT_REQ` and the message returned by the successful completion of the `ioctl` is equivalent to the TI message type `T_OPTMGMT_ACK`.

DIAGNOSTICS

If the `ioctl` system call returns with a value greater than 0, the lower 8 bits of the return value will be one of

the TI error codes as defined in <sys/tiuser.h>. If the TI error is of type TSYSERR, then the next 8 bits of the return value will contain an error as defined in <sys/errno.h> [see intro_prm2(2)].

FILES

<sys/timod.h>
<sys/tiuser.h>
<sys/tihdr.h>
<sys/errno.h>

SEE ALSO

tirdwr(7) .
Programmer's Guide: STREAMS
Programmer's Guide: Networking Interfaces

System Administrator's Reference Manual

tirdwr(7)

NAME

`tirdwr` - Transport Interface read/write interface STREAMS module

DESCRIPTION

`tirdwr` is a STREAMS module that provides an alternate interface to a transport provider which supports the Transport Interface (TI) functions of the Network Services library. This alternate interface allows a user to communicate with the transport protocol provider using the `read(2)` and `write(2)` system calls. The `putmsg(2)` and `getmsg(2)` system calls may also be used. However, `putmsg` and `getmsg` can only transfer data messages between user and stream.

The `tirdwr` module must only be pushed [see `I_PUSH` in `streamio(7)`] onto a stream terminated by a transport protocol provider which supports the TI. After the `tirdwr` module has been pushed onto a stream, none of the Transport Interface functions can be used. Subsequent calls to TI functions will cause an error on the stream. Once the error is detected, subsequent system calls on the stream will return an error with `errno` set to `EPROTO`.

The following are the actions taken by the `tirdwr` module when pushed on the stream, popped [see `I_POP` in `streamio(7)`] off the stream, or when data passes through it.

push When the module is pushed onto a stream, it will check any existing data destined for the user to ensure that only regular data messages are present. It will ignore any messages on the stream that relate to process management, such as messages that generate signals to the user processes associated with the stream. If any other messages are present, the `I_PUSH` will return an error with `errno` set to `EPROTO`.

write

The module will take the following actions on data that originated from a `write` system call:

- All messages with the exception of messages that contain control portions (see the `putmsg` and `getmsg` system calls) will be transparently passed onto the module's downstream neighbor.
- Any zero length data messages will be freed by the module and they will not be passed onto the module's downstream neighbor.
- Any messages with control portions will generate an error, and any further system calls associated with the stream will fail with `errno` set to `EPROTO`.

read The module will take the following actions on data that originated from the transport protocol provider:

- All messages with the exception of those that contain control portions (see the `putmsg` and `getmsg` system calls) will be transparently passed onto the module's upstream neighbor.
- The action taken on messages with control portions will be as follows:
  - Messages that represent expedited data will generate an error. All further system calls associated with the stream will fail with `errno` set to `EPROTO`.
  - Any data messages with control portions will have the control portions removed from the message prior to passing the message on to the upstream neighbor.
  - Messages that represent an orderly release indication from the transport provider will generate a zero length data message, indicating the end of file, which will be sent to the reader of the stream. The orderly release message itself will be freed by the module.
  - Messages that represent an abortive disconnect indication from the transport provider will cause all further `write` and `putmsg` system calls to fail with `errno` set to `ENXIO`. All further `read` and `getmsg` system calls will return zero length data (indicating end of file) once all previous data has been read.
  - With the exception of the above rules, all other messages with control portions will generate an error and all further system calls associated with the stream will fail with `errno` set to `EPROTO`.

- Any zero length data messages will be freed by the module and they will not be passed onto the module's upstream neighbor.

pop When the module is popped off the stream or the stream is closed, the module will take the following action:

- If an orderly release indication has been previously received, then an orderly release request will be sent to the remote side of the transport connection.

SEE ALSO

`getmsg(2)`, `intro_prm2(2)`, `putmsg(2)`, `read(2)`, `write(2)`, `streamio(7)`, `timod(7)`.

Programmer's Guide: STREAMS

Programmer's Guide: Networking Interfaces

System Administrator's Reference Manual

ttcompat(7)

NAME

ttcompat - V7, 4BSD and XENIX STREAMScompatibility module

SYNOPSIS

```
#include <sys/stream.h>
#include <sys/stropt.h>
#include <sys/ttcompat.h>
#include <sys/ttold.h>
ioctl(fd, I_PUSH, "ttcompat");
```

DESCRIPTION

ttcompat is a STREAMS module that translates the ioctl calls supported by the older Version 7, 4BSD and XENIX terminal drivers into the ioctl calls supported by the termio interface [see termio(7)]. All other messages pass through this module unchanged; the behavior of read and write calls is unchanged, as is the behavior of ioctl calls other than the ones supported by ttcompat.

This module can be automatically pushed onto a stream using the autopush mechanism when a terminal device is opened; it does not have to be explicitly pushed onto a stream. This module requires that the termios interface be supported by the modules and the application can push the driver downstream. The TCGETS, TCSETS, and TCSETSF ioctl calls must be supported; if any information set or fetched by those ioctl calls is not supported by the modules and driver downstream, some of the V7/4BSD/XENIX functions may not be supported. For example, if the CBAUD bits in the c_cflag field are not supported, the functions provided by the sg_ispeed and sg_ospeed fields of the sgttyb structure (see below) will not be supported. If the TCFLSH ioctl is not supported, the function provided by the TIOCFLUSH ioctl will not be supported. If the TCXONC ioctl is not supported, the functions provided by the TIOCSTOP and TIOCSTART ioctl calls will not be supported. If the TIOCMBIS and TIOCMBIC ioctl calls are not supported, the functions provided by the TIOCSDTR and TIOCCDTR ioctl calls will not be supported.

The basic ioctl calls use the sgttyb structure defined by <sys/ioctl.h>:

```
struct sgttyb
{
    char sg_ispeed;
    char sg_ospeed;
    char sg_erase;
    char sg_kill;
    int  sg_flags;
};
```

The sg_ispeed and sg_ospeed fields describe the input and output speeds of the device, and reflect the values in the c_cflag field of the termios structure. The sg_erase and sg_kill fields of the argument structure specify the erase and kill characters respectively, and reflect the values in the VERASE and VKILL members of the c_cc field of the termios structure.

The sg_flags field of the argument structure contains several flags that determine the system's treatment of the terminal. They are mapped into flags in fields of the terminal state, represented by the termios structure.

Delay type 0 is always mapped into the equivalent delay type 0 in the c_oflag field of the termios structure. Other delay mappings are performed as follows:

```
sg_flags
     c_oflag □
BS1  BS1
FF1  VT1
CR1  CR2
```

CR2 CR3

CR3 not supported

TAB1 TAB1

TAB2 TAB2

XTABS
TAB3

NL1 ONLRET|CR1

NL2 NL1

If previous TIOCLSET or TIOCLBIS ioctl calls have not selected LITOUT or PASS8 mode, and if RAW mode is not selected, the ISTRIP flag is set in the c_iflag field of the termios structure, and the EVENP and ODDP flags control the parity of characters sent to the terminal and accepted from the terminal:

- Parity is not to be generated on output or checked on input; the character size is set to CS8 and the flag is cleared in the c_cflag field of the termios structure.
- Even parity characters are to be generated on output and accepted on input; the flag is set in the c_iflag field of the termios structure, the character size is set to CS7 and the flag is set in the c_cflag field of the termios structure.
- Odd parity characters are to be generated on output and accepted on input; the flag is set in the c_iflag field, the character size is set to CS7 and the and flags are set in the c_cflag field of the termios structure.
- Even parity characters are to be generated on output and characters of either parity are to be accepted on input; the flag is cleared in the c_iflag field, the character size is set to CS7 and the flag is set in the c_cflag field of the termios structure.

The RAW flag disables all output processing (the OPOST flag in the c_oflag field, and the XCASE flag in the c_lflag field, are cleared in the termios structure) and input processing (all flags in the c_iflag field other than the IXOFF and IXANY flags are cleared in the termios structure). 8 bits of data, with no parity bit, are accepted on input and generated on output; the character size is set to CS8 and the PARENB and PARODD flags are cleared in the c_cflag field of the termios structure. The signal-generating and line-editing control characters are disabled by clearing the ISIG and ICANON flags in the c_lflag field of the termios structure.

The CRMOD flag turns input RETURN characters into NEWLINE characters, and output and echoed NEWLINE characters to be output as a RETURN followed by a LINEFEED. The ICRNL flag in the c_iflag field, and the OPOST and ONLCR flags in the c_oflag field, are set in the termios structure.

The LCASE flag maps uppercase letters in the ASCII character set to their lowercase equivalents on input (the IUCLC flag is set in the c_iflag field), and maps lowercase letters in the ASCII character set to their uppercase equivalents on output (the OLCUC flag is set in the c_oflag field). Escape sequences are accepted on input, and generated on output, to handle certain ASCII characters not supported by older terminals (the XCASE flag is set in the c_lflag field).

Other flags are directly mapped to flags in the termios structure:

sg_flags
flags in termios structure □

CBREAK
complement of ICANON in c_lflag field

ECHO ECHO in c_lflag field

TANDEM
IXOFF in c_iflag field

Another structure associated with each terminal specifies characters that are special in both the old Version 7 and the newer 4BSD terminal interfaces. The following structure is defined by <sys/ioctl.h>:

```
struct tchars
{
```

```
    char t_intrc;   /* interrupt */
    char t_quitc;   /* quit */
    char t_startc;  /* start output */
    char t_stopc;   /* stop output */
    char t_eofc;    /* end-of-file */
    char t_brkc;    /* input delimiter (like nl) */
};
```

XENIX defines the `tchar` structure as `tc`. The characters are mapped to members of the `c_cc` field of the `termios` structure as follows:

`tchars`
: `c_cc` index □

`t_intrc`
: `VINTR`

`t_quitc`
: `VQUIT`

`t_startc`
: `VSTART`

`t_stopc`
: `VSTOP`

`t_eofc`
: `VEOF`

`t_brkc`
: `VEOL`

Also associated with each terminal is a local flag word, specifying flags supported by the new 4BSD terminal interface. Most of these flags are directly mapped to flags in the `termios` structure:

local flags
: flags in `termios` structure □

`LCRTBS`
: not supported

`LPRTERA`
: `ECHOPRT` in the `c_lflag` field

`LCRTERA`
: `ECHOE` in the `c_lflag` field

`LTILDE`
: not supported

`LTOSTOP`
: `TOSTOP` in the `c_lflag` field

`LFLUSHO`
: `FLUSHO` in the `c_lflag` field

`LNOHANG`
: `CLOCAL` in the `c_cflag` field

`LCRTKIL`
: `ECHOKE` in the `c_lflag` field

`LCTLECH`
: `CTLECH` in the `c_lflag` field

`LPENDIN`
: `PENDIN` in the `c_lflag` field

`LDECCTQ`
: complement of `IXANY` in the `c_iflag` field

`LNOFLSH`
: `NOFLSH` in the `c_lflag` field

Another structure associated with each terminal is the `ltchars` structure which defines control characters for the new 4BSD terminal interface. Its structure is:

```
struct ltchars
{
    char t_suspc;   /* stop process signal */
    char t_dsuspc;  /* delayed stop process signal */
    char t_rprntc;  /* reprint line */
    char t_flushc;  /* flush output (toggles) */
    char t_werasc;  /* word erase */
    char t_lnextc;  /* literal next character */
};
```

The characters are mapped to members of the `c_cc` field of the `termios` structure as follows:

`ltchars`
: `c_cc` index □

`t_suspc`
: `VSUSP`

`t_dsuspc`
: `VDSUSP`

`t_rprntc`
: `VREPRINT`

`t_flushc`
: `VDISCARD`

`t_werasc`
: `VWERASE`

`t_lnextc`
: `VLNEXT`

IOCTLS

`ttcompat` responds to the following `ioctl` calls. All others are passed to the STREAMS module below.

`TIOCGETP`
: The argument is a pointer to an `sgttyb` structure. The current terminal state is fetched; the appropriate characters in the terminal state are stored in that structure, as are the input and output speeds. The values of the flags in the `sg_flags` field are derived from the flags in the terminal state and stored in the structure.

`TIOCEXCL`
: Set "exclusive-use" mode; no further opens are permitted until the file has been closed.

`TIOCNXCL`
: Turn off "exclusive-use" mode.

`TIOCSETP`
: The argument is a pointer to an `sgttyb` structure. The appropriate characters and input and output speeds in the terminal state are set from the values in that structure, and the flags in the terminal state are set to match the values of the flags in the `sg_flags` field of that structure. The state is changed with a `TCSETSF` `ioctl` so that the interface delays until output is quiescent, then throws away any unread characters, before changing the modes.

`TIOCSETN`
: The argument is a pointer to an `sgttyb` structure. The terminal state is changed as `TIOCSETP` would change it, but a `TCSETS` `ioctl` is used, so that the interface neither delays nor discards input.

`TIOCHPCL`
: The argument is ignored. The `HUPCL` flag is set in the `c_cflag` word of the terminal state.

`TIOCFLUSH`
: The argument is a pointer to an `int` variable. If its value is zero, all characters waiting in input or output queues are flushed. Otherwise, the value of the `int` is treated as the logical OR of the `FREAD` and

FWRITE flags defined by <sys/file.h>; if the FREAD bit is set, all characters waiting in input queues are flushed, and if the FWRITE bit is set, all characters waiting in output queues are flushed.

TIOCBRK
: The argument is ignored. The break bit is set for the device.

TIOCCBRK
: The argument is ignored. The break bit is cleared for the device.

TIOCSDTR
: The argument is ignored. The Data Terminal Ready bit is set for the device.

TIOCCDTR
: The argument is ignored. The Data Terminal Ready bit is cleared for the device.

TIOCSTOP
: The argument is ignored. Output is stopped as if the STOP character had been typed.

TIOCSTART
: The argument is ignored. Output is restarted as if the START character had been typed.

TIOCGETC
: The argument is a pointer to a tchars structure. The current terminal state is fetched, and the appropriate characters in the terminal state are stored in that structure.

TIOCSETC
: The argument is a pointer to a tchars structure. The values of the appropriate characters in the terminal state are set from the characters in that structure.

TIOCLGET
: The argument is a pointer to an int. The current terminal state is fetched, and the values of the local flags are derived from the flags in the terminal state and stored in the int pointed to by the argument.

TIOCLBIS
: The argument is a pointer to an int whose value is a mask containing flags to be set in the local flags word. The current terminal state is fetched, and the values of the local flags are derived from the flags in the terminal state; the specified flags are set, and the flags in the terminal state are set to match the new value of the local flags word.

TIOCLBIC
: The argument is a pointer to an int whose value is a mask containing flags to be cleared in the local flags word. The current terminal state is fetched, and the values of the local flags are derived from the flags in the terminal state; the specified flags are cleared, and the flags in the terminal state are set to match the new value of the local flags word.

TIOCLSET
: The argument is a pointer to an int containing a new set of local flags. The flags in the terminal state are set to match the new value of the local flags word.

TIOCGLTC
: The argument is a pointer to an ltchars structure. The values of the appropriate characters in the terminal state are stored in that structure.

TIOCSLTC
: The argument is a pointer to an ltchars structure. The values of the appropriate characters in the terminal state are set from the characters in that structure.

FIORDCHK
: FIORDCHK returns the number of immediately readable characters. The argument is ignored.

FIONREAD
: FIONREAD returns the number of immediately readable characters in the int pointed to by the argument.

LDSMAP
: Calls the function emsetmap(*tp*, *mp*) if the function is configured in the kernel.

LDGMAP
: Calls the function emgetmap(*tp*, *mp*) if the function is configured in the kernel.

`LDNMAP`

Calls the function `emunmap`(*tp*, *mp*) if the function is configured in the kernel.

The following `ioctls` are returned as successful for the sake of compatibility. However, nothing significant is done (i.e. the state of the terminal is not changed in any way).

```
TIOCSETD     LDOPEN
TIOCGETD     LDCLOSE
DIOCSETP     LDCHG
DIOCSETP     LDSETT
DIIOGETP     LDGETT
```

NOTES

`TIOCBRK` and `TIOCCBRK` should be handled by the driver. `FIONREAD` and `FIORDCHK` are handled in the stream head.

SEE ALSO

`ioctl`(2), `termios`(3C), `ldterm`(7), `termio`(7).

System Administrator's Reference Manual

tty(7)

NAME

`tty` - controlling terminal interface

DESCRIPTION

The `/dev/tty` file is, in each process, a synonym for the control terminal associated with the process group of that process, if any. It is useful for programs or shell sequences that wish to be sure of writing messages on the terminal no matter how output has been redirected. It can also be used for programs that demand the name of a file for output, when typed output is desired and it is tiresome to find out what terminal is currently in use.

FILES

`/dev/tty`
`/dev/tty*`

SEE ALSO

`console(7)`, `ports(7)`.

System Administrator's Reference Manual

ubft(7)
RM600 only

NAME

`ubft` - monitors power supply, temperature of the environment,air and other hardware components

DESCRIPTION

The purpose of the `ubft` driver is to provide information on monitored hardware and to make the `ubft` daemon available to evaluate this information, and if necessary, initiate reactions such as shutting down the system. Monitored hardware includes fans and fan covers, power packs, BBUs (Battery Backup Units) of various types, and UPS (Uninterruptible Power Supply).

Changes in the status of BBU, fans, and temperature are detected by the EIP (Environment Interface Processor) and reported to the boot controller CSI (Central Services & Interface Processor) or to the (E)HIOS (High Performance Input/Output Subsystem). The boot controller reports status changes to the `ubft` driver by means of MultibusII alarm messages. The driver then sends the `SIGUSR1` signal to the `ubft` daemon.

Only one `ubft` daemon can be active at any time. Activation of other daemons is prevented. Once the `ubft` daemon is terminated, the `ubft` driver cannot initiate any more shutdowns. In order to ensure that the `ubft` daemon does not terminate unintentionally, it can only be shut down with superuser authorization and the `SIGKILL` signal.

Error messages from the `ubft` daemon and the `ubft` driver are reported by the `ubft` driver using the `drv_err` routine.

The reason for a shutdown initiated by the `ups` daemon can be read from the kernel log file the next time the system is brought up.

NOTES

When connecting an external BBU or UPS, the system cannot check whether all cabinets in the network are being correctly supplied by them, and whether the signal connection is linked properly. The system engineer is therefore responsible for ensuring that the connections are set up and configured correctly.

If hardware malfunctions occur, the EIP activates a so-called "auto power off counter" that automatically renders a cabinet or the system powerless after a certain time delay. This means that it is also possible to switch the system off when the `ubft` daemon is not running. This can result in loss of data since this is not a correct shutdown from the point of view of Reliant UNIX.

FILES

`/dev/ubft`
: Device driver of the `ubft` driver (character device, major 48, minor 0).

`/sbin/ups`
: Command for starting and querying the `ubft` daemon.

`/usr/include/sys/io/ubft.h`
: Description of the `ioctl` calls.

`/var/adm/ubft/_upstab`
: Parameterization file for the `ubft` daemon.

SEE ALSO

`ups(1M)`, `_upstab(4)`, `user_reaction(7)`.

System Administrator's Reference Manual

user_reaction(7)
RM600 only

NAME

`user_reaction -` user reaction program interface

SYNOPSIS

`/etc/opt/havail/user_reaction` *eventtype eventstate cabno time* [*item*]

DESCRIPTION

`user_reaction` is a user reaction program interface. It has a program of the same name that can be written by the system administrator so that high availability configurations in the form of cluster systems can be optimally supported. There is no user reaction program on the system by default.

The user reaction program can be called by various software components. It is called from the UBFT daemon by default if the UBFT daemon detected a cabinet monitoring event and a user reaction program exists.

PARAMETERS

The following section describes the call parameters as they are used by the UBFT daemon.

eventtype

This parameter consists of one of the `FANFAIL`, `TEMPFAIL`, `POWERFAIL` (error event), or `INFO` (information event) character strings.

`FANFAIL`
An error occurred in a non redundant fan. Cooling of the cabinet components is no longer guaranteed.

`TEMPFAIL`
The temperature of the air circulating in the cabinet is too high.

`POWERFAIL`
A power failure occurred.

`INFO` An information event occurred. These events are not created by default, but can be configured in the `_upstab(4)` parameterization file of the UBFT daemon.

eventstate

This parameter consists of one of the `OCCURRED`, `EXISTING`, or `DISAPPEARED` character strings.

`OCCURRED`
An error event or information event occurred.

`EXISTING`
An error still exists after expiration of the user time. This event state is not created by default, but can be configured in the `_upstab(4)` parameterization file of the UBFT daemon.

`DISAPPEARED`
An error event previously reported using `OCCURRED` no longer exists.

cabno

This parameter specifies the number of the cabinet for which the event is reported. In systems with EIP2 master, the cabinet number corresponds to the logical cabinet ID from the range [0,15]. In systems with EIP3 master, the cabinet number corresponds to the physical cabinet ID from the range [0,63].

time This parameter specifies the remaining time in seconds until the start of system shutdown when an error event occurs. In information events, it always contains the number `0`.

item This parameter consists of an optional character string that can be configured for each error event and information event and for each cabinet in the `_upstab(4)` parameterization file. The character string does not contain any colons. This parameter is **not** specified by the UBFT daemon by default.

NOTES

The user reaction program must have execution rights and can be a shell script or an executable program.

Before the user reaction program is called by the UBFT daemon, it creates an empty shutdown identification file in case of an error event. The name of the shutdown identification file contains the event type and the number of the relevant cabinet:

`/etc/opt/havail/shutdown_`*eventtype*`.PBAY`*cabno*

eventtype and *cabno* correspond to the call parameters of the user reaction program. Before the shutdown process is initiated, the UBFT daemon checks whether the shutdown identification file still exists. If so, the shutdown process is started.

The shutdown identification file can be deleted using a command in the user reaction program. By doing this, system shutdown for error events in specific special cabinets can be prevented.

APPLICATION EXAMPLE

If two peripherals cabinets with mirrored disks are being used simultaneously by two systems and an UBFT error event occurs in one of the peripherals cabinets, the error results in the user reaction program being called by the UBFT systems. Since the systems can, however, continue to work with the correctly functioning peripherals cabinet, shutdown can be prevented on both systems by deleting the relevant shutdown identification files and the defective cabinet can then be switched off from both systems.

The user reaction program can be called a number of times by the UBFT daemon for a cabinet if different error events occur. There are a number of error events for the `POWERFAIL` error event class that can be different depending on the configuration of the unique *item* character string.

SEE ALSO

`ups`(1M), `_upstab`(4), `ubft`(7).

System Administrator's Reference Manual

vdisk(7)

NAME

`vdisk, mirror, mroot` - virtual disk drivers

DESCRIPTION

Note:

Note the new Vdisk Lite administration concept [VL, see `vl(7)`]. This new concept allows virtual disks (vdisks) to be configured via WSA and clearly safeguards against bad configurations.

Virtual disks function identically to traditional physical disks, but their relation to physical disks is determined from a mapping of a physical disk (or disks) to a virtual disk. This mapping is performed by the `dkconfig(8)` command on the basis of the system database for Vdisk Lite as well as the `/etc/dktab` file, which is a special configuration file for virtual disks. From the perspective of a user program, the interface to a virtual disk partition is indistinguishable from that of a physical disk. Both the block and the character device disk interfaces are supported.

The virtual disk facility allows you to:

- Create disk partitions through concatenation. This means that file systems up to 4 Gbytes (ufs) or 16 Tbytes (vxfs) in size can be configured, even though disks with a capacity of only 2 Gbytes are being used.
- Create disk partitions through striping that evenly spread file access across several disk drives and/or controllers, resulting in average throughput several times greater than that theoretically possible with traditional file system configurations.
- Create any number of disk partitions of arbitrary size (max. 16 Tbytes) without modifying existing partition tables, or reconfiguring the kernel in any way.
- Configure mirrored disks to increase data availability. The system disk can also be mirrored which increases system availability.
- Configure "disks" in the main memory of the system.

The virtual disk facility does not change the existing disk partitioning scheme in any way; both the existing physical disk partition facility and the virtual disk facility may be used together without incompatibility.

Virtual disks provide a level of disk access not directly tied to a particular disk drive, controller, or I/O controller subsystem. A virtual disk may span several disk drives (of any type) and/or several controllers (of any type). A virtual disk may even be a piece of another virtual disk. In general, any supported block device can be a piece of a virtual disk.

"vdisk" is implemented as a new layer between the lowest level of the kernel and the highest level of the I/O drivers.

Virtual disks are administered with the `dkconfig` command and the system database for Vdisk Lite as well as the `/etc/dktab` file. This administration concept corresponds largely to the administration of mounted file systems with `mount(1M)` and `umount`(1M) in accordance with the specifications in `/etc/vfstab`.

VIRTUAL DISK TYPES (WITHOUT VL-VDISKS)

There are different types of virtual disks supported by the "vdisk" product, as follows:

`simple`

A "simple" virtual disk consists of a single piece of a single physical disk drive. In this respect, a simple virtual disk is identical to the notion of a disk partition. An offset and length may be optionally specified that allows one to further subdivide an existing physical disk partition. If the `s7` partition is used (which is conventionally the entire user-accessible portion of a single disk), one can define new and arbitrary partitions independent of the hard-coded partition tables configured in the kernel. This allows sorted disk partitioning, without restrictions to names or number of partitions. It can be used, for example, to create many small partitions on a single disk or within a single disk partition.

This implies that all disk partitions can be created using virtual disks. The only reference to a physical

disk in the kernel-based partition tables would be for the `s7` partition. All further partitioning would be defined with virtual disks in `/etc/dktab`.

`concat`

A "concat" virtual disk consists of two or more pieces of one or more physical disk drives. However pieces can also be virtual disks. It is the logical sum of two or more disk partitions. This is the reverse of the notion of partitioning a disk into smaller sections. Instead, one can combine disks into one large logical disk.

A virtual "concat" type disk can theoretically be 2^{64} = 16 Ebytes in size. A vxfs file system can be 2^{44} = 16 Tbytes maximum, while a ufs file system can be 4 Gbytes maximum. (In all three cases, the upper limit is minus 1 block.) All I/O to a virtual disk is identical to I/O on a physical disk of a capacity equal to the sum of the pieces of the virtual disk. The physical pieces of a concatenated virtual disk may be of any size and in any order. An offset and length may be optionally specified that allows one to further subdivide an existing physical disk partition. Thus, it is possible to concatenate any mix of entire partitions and/or portions of partitions.

`stripe`

A "stripe" virtual disk, like a concatenated virtual disk, also consists of two or more pieces of one or more physical disks. However, instead of the composition being the logical sum of each piece, an interleave algorithm is employed in translating block numbers of the virtual disk into those of the physical disks. This allows sequential I/O on the virtual disk to be translated into interleaved I/O across two or more physical disks.

For striped virtual disks, each piece must be of equal size. An offset and length may be optionally specified that allows one to further subdivide an existing physical disk partition.

The cluster size of a striped partition determines how the blocks of sequential I/O are distributed among the two or more pieces of the physical disk. Each striped disk consists of pieces, each piece is made up of one or more clusters, and each cluster is an integral number of disk blocks in length. The interleave algorithm proceeds sequentially by accessing the first cluster of the first piece, the first cluster of the second piece, ..., the first cluster of the Nth piece, the second cluster of the first piece, ..., and finally the Nth cluster of the Nth piece.

The reason for striping is that, in some circumstances, far greater I/O throughput is achieved in striping mode than in normal mode. For example, if user programs do a massive number of I/Os, the I/Os can be distributed evenly across several disk drives or controllers instead of all confined to one drive.

Note:

In a boundary case of N disks, all of size M and a cluster size equal to M, there is no I/O throughput advantage to using striping as opposed to concatenation.

`memory`, `vmemory`

Virtual disks of the type "memory" or "vmemory" use part of the main memory of the system as their storage medium. Their advantages over physical disks are high access and transfer speeds. They can be used to store data or programs that are accessed frequently. "memory" disks can be used as character-oriented or block-oriented devices, and applications cannot distinguish between them and physical disks. The contents of a "memory" disk are lost if the system is shut down or if it crashes. To avoid losing data, it is possible to mirror "memory" disks with physical (or virtual) disks.

Note:

Virtual disks of type "memory" must no longer be configured using bootflags as of V5.42. If, however, bootflags are still used, the system may crash (panic).

`statesave`, `statsv`

Statesave devices are used exclusively for nonvolatile storage of the status of mirror disks. During configuration, virtual disks can be divided into groups. "statesave" virtual disks are used to store the status of all mirror disks of a particular group, while "statsv" virtual disks perform the same function for non-grouped mirror disks. Only one "statesave" virtual disk is required for a group, and one "statsv" virtual disk for all non-grouped mirror disks.

For further details, please refer to the manual entitled *Virtual Disks*.

`mirror`
: See the next section.

MIRRORED VIRTUAL DISKS

Mirror disk (type "mirror") is a Reliant UNIX facility for enhanced reliability of the disk storage subsystem through duplication of all I/O to two or more physical devices. With two copies of the same data on different disks, should one disk fail, the system can continue to operate reliably without interruption.

However, an increase in the number of drives and possibly controllers may be justified due to the increased storage demands of data replication.

A complete description of the Mirror Disk facility, including installation, configuration, and operation may be found in the *Virtual Disks* manual.

A write operation to the mirror disk is translated into a write to each of the pieces. This guarantees that each piece contains exactly the same data. A read operation from the mirror disk is translated into a read from just one of the pieces. Since all pieces contain exactly the same data, it doesn't matter which piece is chosen for the read. In practice, the piece that can do the read in the least amount of time is chosen. This, and the fact that the mirror disk pieces reside on different disk drives, allows the net performance of mirror disk reads to exceed that of non-mirrored disks.

Read errors on one piece of a mirror disk are retried on another piece. As long as a read can be done successfully from one piece, the mirror read is considered error-free. Unrecoverable or hard write errors deactivate the piece on which the error occurred, but as long as one piece does the write successfully, the mirror write is considered error-free. The user process doing the mirror disk I/O is unaware of any error handling as long as the I/O is successful on one piece.

Mirror Disk State

A mirror disk piece can be in one of four states: `online`, `disabled`, `enabled`, or `Master`.

`online`
: This is the normal state for a mirror disk piece. An `online` piece has a completely valid copy of the mirror disk data and can be used for reading and writing by the user.

`disabled`
: When there is an unrecoverable I/O error on an `online` piece, the piece is automatically `disabled`. No further user I/O is attempted on a `disabled` piece since the piece is no longer trusted to have a valid copy of the mirror disk data. The piece remains `disabled` until the administrator manually changes its state. `disabled` is also the default state when a mirror disk piece is configured for the first time.

`enabled`
: When it is desired to put a `disabled` mirror disk piece back in service, it is `enabled`. This state makes the piece eligible for use again, but it will not be promoted to `online` until a fresh copy of the mirror disk data has been copied to the piece. This copying is accomplished automatically by the initiation of a system process that reads valid data from `online` pieces and writes it to `enabled` pieces. Then the `enabled` piece is promoted to the `online` state since it is now known to have an identical copy of the mirror disk data.

`Master`
: Corresponds to `enabled`; however this piece is defined as the master piece. This piece is set to `online` during the next configuration process and acts as a source for the copying process.

Since each mirror disk piece has a state, the mirror virtual disk itself has state:

`MIRRORED`
: A mirror disk is `MIRRORED` when at least two pieces are `online`. In this state the user of the mirror disk may be immune to all single point failures.

`NOT-MIRRORED`
: A mirror disk is `NOT-MIRRORED` if only one piece is `online`. The mirror disk is still perfectly usable, but

there is no redundancy.

`DOWN` A mirror disk is `DOWN` when there are no `online` pieces. No user I/O can be done to a mirror disk that is `DOWN`.

State changes occur either automatically or manually.

Mirror Disk Catchup

Integral to the mirror disk driver is the capability to synchronize the pieces of the mirror disk. This is required either initially when a mirror disk is first configured, or later if a mirror disk piece is ever `disabled`. The synchronization is accomplished by copying data from `online` pieces to `enabled` pieces. When the copy has completed, the `enabled` pieces are promoted to the `online` state. This process of copying is called "catchup". The mirror disk may be in use while catchup is running. For example, you may run `fsck` on a mirror disk while catchup is running.

Catchup I/O does not interfere with user I/O.

A catchup begins automatically any time there is at least one `online` piece (which will be the source of the copy) and at least one `enabled` piece (which will be the destination). The mirror disk driver initiates the catchup process, and if it completes successfully, all `enabled` pieces are put `online`.

Configuring a mirror disk with /etc/dktab

The next example shows a possible entry for a mirror disk in `/etc/dktab`:

```
/dev/vd/vdisk1        mirror      2
      /dev/ios0/sdisk002s1
      /dev/ios0/sdisk003s1
```

This defines `/dev/vd/vdisk1` as a mirrored virtual disk with 2 pieces. Piece 1 is `sdisk002s1` and piece 2 is `sdisk003s1`.

An optional offset and length may also be given, such as:

```
/dev/vd/vdisk1        mirror      2
      /dev/ios0/sdisk002s1        0        74358
      /dev/ios0/sdisk003s1        0        74358
```

In order to use mirror disks, a virtual disk of the type "statsv" or "statesave" is required. These virtual disks can consist of one or more pieces, each containing status information on the mirror disks assigned to it. If a disk comprises a number of pieces, all pieces contain the same information. The size of a "statsv" or "statesave" virtual disk must be at least 6 Kbytes. The following is an example of a "statsv" virtual disk with two pieces. Offset and size are optional.

```
/dev/vd/vdisk0        statsv      2
      /dev/ios0/sdisk002s1        0        12
      /dev/ios0/sdisk003s1        0        12
```

There can only be one virtual disk of the "statsv" type in a system. Alternatively, virtual disks of the "statesave" type can also be used. This type of virtual disk records the status information of a particular group of mirror disks. Example:

```
/dev/vd/vdisk0        statesave,group=0      2
        /dev/ios0/sdisk002s1        0        32
        /dev/ios0/sdisk003s1        0        32
/dev/vd/vdisk3        mirror,group=0         2
        /dev/ios0/sdisk002s5
        /dev/ios0/sdisk003s5
```

Virtual disks of type "mirror,readonly"

Virtual disks of type `mirror,readonly` are mirror disks which contain unchangeable data. What this means, in particular, is that the individual pieces of this mirror disk cannot contain different data. It is therefore not necessary with such mirror disks to perform a catchup because a piece of the disk is temporarily unavailable (e.g. because an FO segment has failed). In this instance, the piece that is temporarily out of service can be brought `online` again with

`dkmirror –C # /dev/vd/vdisk ...`

Syntax

The new `readonly` keyword can be entered in the `/etc/dktab` file. Possible syntax would be

```
/dev/vd/vdisk35 mirror,readonly 2
 /dev/ios0/sdisk032s6
 /dev/ios0/sdisk062s6
```

or when combined with other options:

```
/dev/vd/vdisk35 mirror,olr,readonly,group=7 2
 /dev/ios0/sdisk033s8
 /dev/ios0/sdisk063s8
```

The `readonly` keyword can also be assigned to other types of virtual disks ("simple", "concat" or "stripe"), though it has no meaning for other types ("statesave", "statsv", "memory"). `readonly` is also not supported in OLR groups.

How it works

Both pieces of an existing mirror disk are `online`. It is obvious from the application or external circumstances that the data on this mirror disk will not change again. This is what happens.

The mirror disk is unconfigured. The `readonly` keyword is added to the declaration in the `/etc/dktab` file. Thus, for example:

```
/dev/vd/vdisk35 mirror,readonly,group=27 2
 /dev/ios0 ...
 /dev/ios0 ...
```

The mirror disk is then reconfigured.

An attempt to write to the read-only mirror disk is rejected with the error `EACCES` (permission denied). If a read error occurs in a piece of the disk during operation, this piece is set to the status `disabled`. When this failed piece is available again, it can be brought `online` using

`dkmirror –C`

A check is performed to see if the pieces match. If they do, the failed piece is set `online` again directly (following a request for confirmation but no catchup). If the check fails, `dkmirror –C` terminates with a warning.

MIRRORING THE ROOT PARTITION

"mroot" is an extension of the "mirror" package. It enables the mirroring of the root partition and the primary swap area – though only using `/etc/dktab`. This considerably increases the availability of the system in the event of root disk failure.

Root and swap-partition mirroring only makes sense if all partitions of the root disk are mirrored. Only then can the system continue operating if the root disk fails.

"mroot" partitions (mirrored root or swap partitions) are configured and managed using the same utilities as with normal mirror disks.

Configuration

"mroot" partitions are special mirror disks. Some restrictions apply here, and additional configuration is required.

After an "mroot" partition has been booted, it cannot be unconfigured. You cannot manually disable all of its pieces. At least one piece must remain online.

With earlier versions of `mroot` it was necessary to change the `/stand` directory to a separate file system (i.e. generate a new file system, copy all files from `/stand` to the new file system and mount the new file system as `/stand`). This is no longer necessary but it is still possible.

One additional parameter of `boot2` must be transferred to Reliant UNIX when starting the system in order to initiate root partition mirroring. With the RM600 and the RM400, this is initiated by editing the

`/etc/default/boot` file.

An entry for both the root and primary swap area must be added to the file `/etc/dktab`. These entries are almost identical to a mirror disk entry, with the addition of the subtype `root` or `swap` in the type field (see **EXAMPLES** below).

Next, the configuration files `/stand/mrconfig.1` and `/stand/mrconfig.2` must be set up using the command `dkconfig –R`. These files are used when booting the system. They are based on the entries in the file `/etc/dktab`. If `/stand` is not a separate file system, the configuration data is also stored in `/mrconfig.1` and `/mrconfig.2`.

The entries for the root file system and the first swap area must be changed in the `/etc/vfstab` file (see **EXAMPLES** below).

The last step is to define the physical piece containing the correct data (masterpiece) for the mirror disks. The data is copied to the other piece the next time the system is booted. This is carried out using the command `dkmirror –m` ... (see **EXAMPLES** below).

EXAMPLES

By convention, all virtual disk nodes are of the form:

`/dev/vd/rvdisk[0-9]*` (for character virtual disks).

and

`/dev/vd/vdisk[0-9]*` (for block virtual disks).

Other device names could be given by renaming or linking. However, note that kernel error messages will still print in the form `vdisk#` where # is the unit number (equal to the minor device number).

Virtual disks are specified in the `/etc/dktab` file. The virtual disks are configured or unconfigured using the `dkconfig` utility, which reads the `/etc/dktab` file. See both `dktab(4)` and `dkconfig(8)` for details.

The following is an example of the use of virtual disks to create a concatenated virtual disk and use it for a file system. This example shows step by step what entries `/etc/dktab` requires.

1. Add the appropriate entry to `/etc/dktab`:

```
/dev/vd/vdisk0  concat 4
    /dev/ios0/sdisk000s1
    /dev/ios0/sdisk001s1
    /dev/ios0/sdisk001s2
    /dev/ios0/sdisk001s3
```

2. Configure the virtual disk:

```
# dkconfig -c /dev/vd/vdisk0
```

3. List the current virtual disk configuration:

```
# dkconfig -l /dev/vd/vdisk0
/dev/vd/vdisk0       301440 blocks `Concatenated' in 4 pieces
```

4. Verbose listing of current virtual disk configuration:

```
# dkconfig -lv /dev/vd/vdisk0
Virtual disk /dev/vd/vdisk0: `Concatenated':
    piece  0  /dev/ios0/sdisk000s1  75360 blocks [0...75359]
    piece  1  /dev/ios0/sdisk001s1  75360 blocks [0...75359]
    piece  2  /dev/ios0/sdisk001s2  75360 blocks [0...75359]
    piece  3  /dev/ios0/sdisk001s3  75360 blocks [0...75359]
/dev/vd/vdisk0       301440 blocks `Concatenated' in 4 pieces
```

5. Invoke `newfs(1M)` to create the virtual disk file system:

```
# newfs /dev/vd/vdisk0
```

6. Mount the new file system:

```
# mount -F ufs /dev/vd/vdisk0 /u8
```

The following example shows the mirroring of all partitions of a root disk. `/dev/ios0/sdisk000` is the root disk; `/dev/ios0/sdisk001` is the disk to be used for mirroring. We are assuming identical disks with identical partition tables.

The partitions `/dev/ios0/sdisk000s15` and `/dev/ios0/sdisk001s15` are implemented in order to record "statesave" information (required as with normal mirror disks).

1. Edit the file `/etc/vfstab`.

   Modify the entries

```
/dev/ios0/sdisk000s0  /dev/ios0/rsdisk000s0  /       ufs  0  yes     rw
/dev/ios0/sdisk000s1  /dev/ios0/rsdisk000s1  -       swap -  -       rw
/dev/ios0/sdisk000s2  /dev/ios0/rsdisk000s2  /opt    vxfs 0  yes     rw
/dev/ios0/sdisk000s3  /dev/ios0/rsdisk000s3  /usr    vxfs 0  yes     rw
/dev/ios0/sdisk000s4  /dev/ios0/rsdisk000s4  /var    vxfs 0  yes     rw
/dev/ios0/sdisk000s5  /dev/ios0/rsdisk000s5  /home   vxfs 0  yes     rw
```

   in

```
/dev/root            /dev/rroot              /       ufs  0  yes     rw
/dev/swap            /dev/rswap              -       swap -  -       rw
/dev/vd/vdisk2       /dev/vd/rvdisk2         /opt    vxfs 0  yes     rw
/dev/vd/vdisk3       /dev/vd/rvdisk3         /usr    vxfs 0  yes     rw
/dev/vd/vdisk4       /dev/vd/rvdisk4         /var    vxfs 0  yes     rw
/dev/vd/vdisk5       /dev/vd/rvdisk5         /home   vxfs 0  yes     rw
```

2. Edit the file `/etc/dktab`.

   Define the mirror disks and the "statsv" partition as follows:

```
/dev/vd/vdisk15 statsv 2
      /dev/ios0/sdisk000s15
      /dev/ios0/sdisk001s15
/dev/vd/vdisk0 mirror,root 2
      /dev/ios0/sdisk000s0
      /dev/ios0/sdisk001s0
/dev/vd/vdisk1 mirror,swap 2
      /dev/ios0/sdisk000s1
      /dev/ios0/sdisk001s1
/dev/vd/vdisk2 mirror 2
      /dev/ios0/sdisk000s2
      /dev/ios0/sdisk001s2
/dev/vd/vdisk3 mirror 2
      /dev/ios0/sdisk000s3
      /dev/ios0/sdisk001s3
/dev/vd/vdisk4 mirror 2
      /dev/ios0/sdisk000s4
      /dev/ios0/sdisk001s4
/dev/vd/vdisk5 mirror 2
      /dev/ios0/sdisk000s5
      /dev/ios0/sdisk001s5
/dev/vd/vdisk10 mirror 2
      /dev/ios0/sdisk000s10
      /dev/ios0/sdisk001s10
```

3. Configure the virtual disks for the partitions "statsv", "root", and "swap":

```
# dkconfig -R
```

   Configure the remaining virtual disks: □

```
# dkconfig -vac
```

4. Initialize the status of the new mirror disks, i.e. define the masterpiece:

```
# for i in 0 1 2 3 4 5 10
# do
# dkmirror -d0 -d1 /dev/vd/vdisk$i
# dkmirror -m0 /dev/vd/vdisk$i
# done
```

   Followed by `dkmirror -L` for checking.

5. Define the boot parameters:

   a) for RM600

   Add the following line to the file `/etc/default/boot`:

```
BOOTPx=bootflags=0x40
```

   where `x` is a previously unused number.

   b) for RM400

   Add the following lines to the file `/etc/default/boot`:

```
bootflags=0x40
boot=ios0/sdisk000s0
root=ios0/sdisk000s0
swap=ios0/sdisk000s1
```

6. Reboot the system:

```
# init 6
```

A catchup is started for each mirror disk when the system is booted, i.e. the data is copied from the masterpiece to the other piece.

NOTES

All disk-related statistics, such as those reported by `dkstat(1M)`, are still bound to physical disks. No statistics are gathered based on virtual disks, just as no statistics are gathered based on partitions.

The limitations are 99 pieces to a virtual disk.

The number of virtual disks is a value that can be changed (`MAXVDUNIT`), but must be within the range 1 – 9999 [see `mtune(4)`]; default is 1024.

SEE ALSO

`dktab(4)`, `vl(7)`, `dkconfig(8)`, `dkmirror(8)`, `vdadmin(8)`.

Manual *Virtual Disks*.

System Administrator's Reference Manual

vdisklite(7)

NAME

vdisklite, vl - virtual disk driver for WSA

DESCRIPTION

Vdisk Lite (VL) is a concept that allows virtual disks (vdisks) to be configured under WSA (Web-based System Administration).

Vdisk Lite applies the current vdisk principle [see vdisk(7)] whereby virtual disks can be configured using a command interface [see vdadmin(8)].

The administration information used by Vdisk Lite is stored under /etc/daps/daps in the vdisk.daps and vdiskgrp.daps files. This data is accessed in the framework of cross-system WSA administration.

Since Vdisk Lite is integrated in WSA, the concept allows virtual disks to be configured via a user interface.

Vdisk Lite is a volume manager, i.e.

- free disk areas are managed,
- improved protection is provided against incorrect or inefficient configurations.

The sequence of configuration steps with Vdisk Lite [cf. vdadmin(8)] is strictly defined:

- Create groups
  A new vdisk group must first be created.
- Import disks
  Physical disks are then imported into this group. This operation results in the creation of mirror or base vdisks.
- Create vdisk types
  Various vdisk types can then be created.
- Partition the vdisks
  The resulting vdisks can be partitioned into pieces of the required size (termed "p-vdisks").

Create groups

The following groups can be distinguished:

local
: Disks are only operated from one computer.

  Enabled flag (according to dktab notation): olr (default)

observe
: Not "multi-hosted", OBSERVE configuration.

  Enabled flag (according to dktab notation): olr

  Two different disks in a SCSI port may not be contained in different observe groups.

rms (Passive Shared) "multi-hosted", RMS configuration.
: Enabled flags (dktab notation): olr,hv

ops (Active Shared) "multi-hosted", OPS configuration.
: Enabled flags (according to dktab notation): olr,ops

opsres
: As ops, the only difference being that mirror disks in this group support Oracle resilvering.

  Enabled flags (according to dktab notation): olr,opsres

When a group is created, the associated statesave vdisk is also created. The system administrator has to specify the vdisk name of this statesave vdisk (e.g. vdisk200). This vdisk is only assigned disk space for storing the statesave information when disks are subsequently imported.

Import disks

According to the new concept, all disks are operated on the basis of their position-independent names.

The following nodes exist for each disk:

`/devices/`*serialno*
: # corresponds to `/dev/ios0/sdiskxxxs7`

`/devices/`*rserialno*
: # corresponds to `/dev/ios0/rsdiskxxxs7`

`/devices/`*serialno*`.priv`
: # corresponds to `/dev/ios0/sdiskxxxs15`

`/devices/`*rserialno*`.priv`
: # corresponds to `/dev/ios0/rsdiskxxxs15`

In order to be able to use a disk for vdisks, this disk must first be imported. The specific VL actions when importing a disk are as follows:

1. The partition table is set to `s7` (data area = entire disk apart from one cylinder) and `s15` (private/statesave area = last cylinder). This eliminates any overlapping partitions.
2. A base vdisk is created, which represents the data area (`s7`).
3. This vdisk is assigned to an existing group.
4. This vdisk is entered as available in the `vdisk.daps` file.
5. If this base vdisk is configured with `dkconfig(8)` , the physical disk is disabled for non-vdisk access.
6. It is also possible to create a mirror vdisk directly by importing several disks.

Only disks with synthetic geometry can be imported, i.e. of disk type MP12/MP14 or more recent versions.

Notes:

- The pieces of a complex vdisk must always belong to the same group.
- When a vdisk is created, the pieces it comprises are marked as used in the `vdisk.daps` file and the vdisk itself is marked as available.
- A disk can only be imported if it is regarded as not yet used by the operating system (e.g. not specified in `/etc/vfstab`, `/etc/dktab` etc.).

Create different types of vdisk

The following vdisk types can be distinguished:

Base Vdisks

This is the initial vdisk type created automatically when a physical disk is imported.

A base vdisk always has a 1:1 relationship with its underlying physical disk.

A base vdisk is, in essence, a mirror disk with only one piece. This type of vdisk is therefore also given a statesave entry, which makes it possible to subsequently create mirror vdisks from base vdisks.

Mirror or RAID-1 Vdisks

A mirror vdisk can be created in two ways:

1. Mirrored import
   Two or three disks, which must be the same size, are imported. The partition table is modified for all disks as described above (under "Import disks"). A mirror vdisk is then created whose pieces represent the seventh partitions of these disks.
2. Subsequent mirroring
   A mirror vdisk can be created by adding another base vdisk to a (non-configured) base vdisk. This is best clarified with an example.

   Starting situation:

   ```
   /dev/vd/vdisk35 base
   ```

```
        /devices/00700324666
/dev/vd/vdisk37 base
        /devices/00700324667
```

Result:

```
/dev/vd/vdisk35 mirror
        /devices/00700324666
        /devices/00700324667
```

As one can see, the `vdisk37` disk no longer exists subsequently and the `vdisk35` disk has become a mirror vdisk.

Statesave Vdisks

Statesave vdisks are required to operate mirror disks. They are created automatically when a group is created. When a `ios0/sdiskxyz` disk is imported, a base (or mirror) vdisk is defined by means of the associated `/devices/`*serialno* vdisk and, in addition, the `/devices/`*serialno*`.priv` vdisk (which is actually partition 15 of the disk) is recorded as a statesave piece.

If the statesave already has six pieces, no further pieces will be added. A maximum of two disks are used as statesave pieces for each participating SCSI channel. If a low-level vdisk is deleted, the associated statesave pieces are also removed. This is only permitted if the entire group has been unconfigured. If, following the export, the statesave vdisk comprises two or more pieces, a warning is issued. If the last piece of the statesave vdisk was deleted during the export, the group may not contain any further disks apart from the disk to be exported. Exporting the last disk in a group is not the same as deleting the group.

Stripe or RAID-0 Vdisks

A stripe vdisk is created by applying a tailored algorithm to several (2–99) underlying low-level vdisks. This corresponds to RAID-Level 0.

Concat Vdisks

A concat vdisk is created by logically concatenating either 2 to 99 base or mirror vdisks.

Select Vdisks

The vdisk type "select" is tailored to SRDF configurations of EMC Symmetrix raid boxes.

A select vdisk is a mirror vdisk with special features. It has several pieces, generally two, though only one piece can be accessed at any one time. In the case of SRDF, this would be the R1 side (master side). If the SRDF is switched, the other piece can be made active using a command call. The previously active piece is then deactivated at the same time.

Partition the Vdisks

It is only possible using the previous methods to create vdisks that are at least as large as an entire disk. If you want to create smaller vdisks, you have to partition an existing vdisk (this vdisk is then known as a "volume"). While this is done with disks or RAID-LUNs using `dkpart(8)`, *N* simple vdisks ("p-vdisks") are defined above the volume with this concept. The partitioning hence moves from the disk driver to the vdisk layer. There are no overlapping partitions with this type of partitioning.

You can specify a desired size when defining p-vdisks, although no offset is permitted within the volume.

The p-vdisks must cover the entire volume. If the total of the specified lengths does not correspond to the size of the volume, the remainder of the last p-vdisk is increased.

A previously partitioned volume can be repartitioned if at least one of its p-vdisks is free (marked as available and not configured). The following can happen during repartitioning:

- New p-vdisks can be created on the area of a free p-vdisk.
- Neighboring p-vdisks can be joined. When joining two p-vdisks, the first is extended by the size of the second (entry modified in the `vdisk.daps` file) and the second is deleted from the `vdisk.daps` file. The first p-vdisk may be used but the second must remain unused. This supports the option, for example, of enlarging a Veritas file system.

EXAMPLES

See `vdadmin`(8) .

NOTES

`Vdisk Lite` does not support all vdisk configurations currently possible, for example, `memory` vdisks can still only be configured using `/etc/dktab`.

FILES

`/etc/daps/daps/vdisk.daps`
`/etc/daps/daps/vdiskgrp.daps`
VL configuration files for virtual disks

SEE ALSO

`vdisk`(7) , `vdadmin`(8) .

Manual *Virtual Disks*.

System Administrator's Reference Manual

vxfsio(7)
VxFS

NAME

`vxfsio` - Veritas File System control functions (VxFS)

SYNOPSIS

```
#include <sys/types.h>
#include <sys/fs/vx_ioctl.h>
int ioctl(int fildes, int cmd, arg);
```

DESCRIPTION

The `ioctl(2)` enhancements for VxFS provide for extended control over open files.

The argument *fildes* is an open file descriptor.

The data type and value of *arg* are specific to the type of command specified by *cmd*. Unless specified, *arg* is treated as an `int` type. The symbolic names for commands and file status flags are defined by the `<sys/fs/vx_ioctl.h>` header file.

The enhancements available are:

`VX_SETCACHE`

Set caching advisories. These advisories allow an application to indicate to the file system which forms of caching would be most advantageous. The values for *arg* are such that multiple advisories may be set by combining values with logical OR operations. The possible values for *arg* are:

`VX_RANDOM`

Indicates that the file is being accessed randomly. Read-ahead should not be performed.

`VX_SEQ`

Indicates that the file is being accessed sequentially. Maximum read-ahead should be performed.

`VX_DIRECT`

Indicates that data associated with read and write operations is to be transferred directly to or from the user supplied buffer, without being cached. When this options is enabled, all I/O operations must begin on block boundaries and must be a multiple of the block size in length. The buffer supplied with the I/O operations must be aligned to a page boundary.

If an I/O request fails to meet alignment criteria, or the file is currently being accessed for mapped I/O, the I/O request will be performed as a data synchronous I/O operation.

`VX_NOREUSE`

Indicates that buffered data does not need to be retained in anticipation of further use by the application.

`VX_DSYNC`

Indicates that data synchronous I/O mode is desired. In data synchronous I/O mode, a write operation returns to the caller after the data has been transferred to external media, but the inode is not updated synchronously if only the times in the inode need to be updated.

Only one of `VX_RANDOM`, `VX_SEQ`, or `VX_DIRECT` may be specified. The `VX_NOREUSE` and `VX_DSYNC` options may not be used in conjunction with `VX_DIRECT`. The caching advisories for a file are maintained on a per-file basis. Changes made to the advisories by a process affect I/O operations by all processes currently accessing the file.

The `VX_SETCACHE ioctl` returns a 0 if the caching advisories are successfully set. If the operation fails, the return value is –1 and the external variable `errno` will be a general diagnostic such as the following:

`EINVAL`

The file specified by *fildes* is not a regular file, or the combination of advisories is invalid.

VX_GETCACHE
: Get caching advisories in effect for the file. The argument *arg* should be a pointer to to an int.

  The VX_GETCACHE ioctl returns a 0 if the caching advisories are successfully obtained. If the operation fails, the return value is −1 and the external variable errno will be a general diagnostic such as the following:

  EINVAL
  : The file specified by *fildes* is not a regular file.

VX_SETEXT
: Set extent information. Set the extent information according to the parameters specified by *arg*. The argument *arg* points to a structure of type vx_ext defined in <sys/fs/vx_ioctl.h>, which contains the following members:

```
off_t    ext_size;      /* extent size */
off_t    reserve;       /* space reservation */
int      a_flags;       /* allocation flags */
```

  The ext_size element is used to request a fixed extent size, in blocks, for the file. If a fixed extent size is not required, zero should be used to allow the default allocation policy to be used. Changes to the fixed extent size made after the file contains indirect blocks have no effect unless all current indirect blocks are freed via file truncation and/or reservation deallocation.

  The reserve element is used to set the amount of space preallocated to the file. If the reserve amount is greater than the current reservation, the allocation for the file is increased to match the reserve amount. If the reserve amount is less than the current reservation, the allocation is decreased. The allocation will not be reduced to less than the current file size.

  File reservation cannot be increased beyond the ulimit(2) of the requesting process. However, an existing reservation will not be trimmed to the requesting processes ulimit(2). Reservation of space for existing sparse files will not cause blocks to be allocated to fill in the holes, but will only allocate blocks after the end of the file. Thus, it is possible to have a larger reservation for a file than blocks in the file.

  The reservation amount is independent of file size since reservation is used to preallocate space for a file. The a_flags element is used to indicate the type of reservation required. The choices are:

  VX_NOEXTEND
  : The file may not be extended once the current reservation is exceeded. The reservation may be increased, if necessary, by another invocation of the ioctl, but the file will not be extended automatically.

  VX_TRIM
  : The reservation for the file is to be trimmed to the current file size upon last close by all processes that have the file open.

  VX_CONTIGUOUS
  : The reservation must be allocated contiguously (as a single extent). ext_size will become the fixed extent size for subsequent allocations, but has no affect on this one. The reservation will fail if the file has gone into indirect extents, unless the amount of space requested is the same as the indirect extent size. If the contiguous allocation request is done on an empty file, this will not happen.

  VX_ALIGN
  : Align all new extents on an ext_size boundary relative to the starting block of an allocation unit. If VX_CONTIGUOUS is set, the single extent allocated during this invocation is not subject to the alignment restriction.

  VX_NORESERVE
  : The reservation is to be made as a nonpersistent allocation to the file. The on-disk inode will not updated with the reservation information so that the reservation will not survive a system crash.

The reservation is associated with the file until the close of the file. The reservation is trimmed to the current file size on close.

VX_CHGSIZE
The reservation is to be immediately incorporated into the file. The file's on-disk inode is updated to with size and block count information that is increased to include the reserved space. Unlike an fcntl F_FREESP operation which "truncates-up" [see fcntl(2)], the space included in the file is not initialized.

This operation is restricted to users with appropriate privilege (root).

Write permission to a file is required to set extent information, but any process which can open the file can get the extent information. Extent information only applies to regular files. Only one set of extent information is kept per file. Except in those cases noted above as non-persistent, the extent information becomes part of the on-disk inode information and, thus, persists across reboots. The conditional allocations mentioned above are the only exceptions here (see VX_NORESERVE).

The VX_SETEXT ioctl returns a 0 if the extent information is successfully set. If the operation fails, the return value is –1 and the external variable errno will be a general diagnostic or one of the following:

EINVAL
The file specified by *fildes* is not a regular file, or the requested extent information is invalid.

EACCESS
The calling process does not have write access to the file specified by *fildes*.

EFBIG
Attempting to reserve space larger than the maximum file size limit for this process.

VX_GETEXT
Get extent information. Return the extent information associated with *fildes*. The argument *arg* points to a structure of type vx_ext, as defined in <sys/fs/vx_ioctl.h>.

The VX_GETEXT ioctl returns a 0 if the extent information is successfully obtained. If the operation fails, the return value is –1 and the external variable errno will be a general diagnostic or the following:

EINVAL
The file specified by *fildes* is not a regular file.

VX_GETFSOPT
Get file system options. The argument *arg* should be a pointer to to an int. This command may be used by any user who can open the root inode on the file system.

The options returned in *arg* are:

VX_FSO_BLKCLEAR
Indicates that all newly allocated blocks will be guaranteed to contain all zeros.

VX_FSO_CACHE_CLOSESYNC
Indicates that any nonlogged changes to the inode or data will be flushed to disk when the file is closed.

VX_FSO_CACHE_DIRECT
Indicates that any nonsynchronous I/O will be handled as if the VX_DIRECT cache advisory had been set on the file. Also, any non-logged changes to the inode or data will be flushed to disk when the file is closed.

VX_FSO_CACHE_DSYNC
Indicates that any writes that do not have either O_SYNC or the VX_DIRECT advisory set will be handled as if the VX_DSYNC advisory had been set on the file. Also, any nonlogged changes to the inode or data will be flushed to disk when the file is closed.

VX_FSO_NODATAINLOG
Indicates that intent logging of user data for synchronous writes is disabled.

VX_FSO_NOLOG

Indicates that intent logging of structural changes to the file system is disabled.

VX_FSO_OSYNC_CLOSESYNC
Indicates that any non-logged changes to the inode or data will be flushed to disk when a file accessed with O_SYNC is closed.

VX_FSO_OSYNC_DIRECT
Indicates that any O_SYNC I/O will be handled as if the VX_DIRECT cache advisory had been set on the file instead. Also, any non-logged changes to the inode or data will be flushed to disk when a file accessed with O_SYNC is closed.

VX_FSO_OSYNC_DSYNC
Indicates that any O_SYNC writes will be handled as if the VX_DSYNC cache advisory had been set on the file instead. Also, any non-logged changes to the inode or data will be flushed to disk when a file accessed with O_SYNC is closed.

VX_FSO_SNAPPED
Indicates that a snapshot backup is in progress on the file system.

VX_FSO_SNAPSHOT
Indicates that this file system is a snapshot backup of another file system.

The VX_GETFSOPT ioctl returns a 0 if the file system options are successfully obtained. If the operation fails, the return value is −1 and the external variable errno will be a general diagnostic or the following:

EINVAL
The file specified by *fildes* is not a regular file.

VX_FREEZE
Sync, then freeze the file system. Once frozen, all further operations against the file system block until a VX_THAW operation is received. The argument *arg* is a timeout value expressed in seconds. If a VX_THAW operation is not received within the specified timeout interval, the file system will perform a VX_THAW operation automatically.

This command may only be used by a user with appropriate privilege on the root directory of the file system.

The VX_FREEZE ioctl returns a 0 if the file system is successfully frozen. If the operation fails, the return value is −1 and the external variable errno will be a general diagnostic or the following:

EINVAL
The timeout value specified by *arg* is less than 1.

VX_THAW
Unblock a file system which has been frozen by a VX_FREEZE operation. The argument *arg* should be NULL. The process which is to issue a VX_THAW operation must have the root directory of the file system open, and must ensure that it does not access the file system after the file system has been frozen, to ensure that the process itself does not block.

This command may only be used by a user with appropriate privilege on the root directory of the file system.

The VX_THAW ioctl returns a 0 if the file system is successfully unfrozen. If the operation fails, the return value is −1 and the external variable errno will be a general diagnostic or one of the following:

EAGAIN
The file system is not currently frozen.

VX_EXPAND
Expand the size of the file system. The argument *arg* specifies the desired size, expressed in file system blocks.

This command may only be used by a user with appropriate privilege on the root directory of the file system.

The VX_EXPAND ioctl returns a 0 if the file system is successfully expanded. If the operation fails,

the return value is –1 and the external variable `errno` will be a general diagnostic or one of the following:

`EROFS`
: The file system is mounted read-only.

`EINVAL`
: The requested file system size is not larger than the current file system size, or is larger than the maximum supported file system size.

`VX_DIRSORT`
: Sort and compress a set of directory blocks. The argument *arg* points to a structure of type `dirsort` as defined in `<sys/fs/vx_ioctl.h>`, which contains the following members:

```
#define   VX_DIRSORTBUFS 2             /* number of directory buffers */
ino_t     dir_inode;                   /* inode number of directory */
daddr_t   dir_off[VX_DIRSORTBUFS];     /* block offset in directory */
daddr_t   dir_bno[VX_DIRSORTBUFS];     /* original block numbers */
daddr_t   dir_ibuf[VX_DIRSORTBUFS];    /* original block contents */
daddr_t   dir_obuf[VX_DIRSORTBUFS];    /* new block contents */
daddr_t   dir_olen[VX_DIRSORTBUFS];    /* new length of blocks */
```

The directory inode specified by `dir_inode` will be reorganized. The blocks `dir_blkoff` and `dir_blkoff` + 1 within the directory will be read. If the directory entries in those blocks are identical to the directory entries in `dir_buf`, then the directory entries will be reordered to match the `dir_buf`.

If the reordered directory can fit in immediate data, then it will be placed in the immediate data area and the directory block will be released. If the reordering leaves the last block of the directory free, then the block will be released.

This command may only be used by a user with appropriate privilege on the root directory of the file system.

The `VX_DIRSORT ioctl` returns a 0 if the caching advisories are successfully set. If the operation fails, the return value is –1 and the external variable `errno` will be a general diagnostic or one of the following:

`EINVAL`
: A member of the `dirsort` structure is invalid.

`ENOTDIR`
: The file specified by `dir_inode` is not a directory.

`EBUSY`
: The current directory contents do not match those specified in the `dirsort` structure.

`VX_EXTREORG`
: Reorganize a set of extents in a file. The argument *arg* points to a structure of type `extreorg`, as defined in `<sys/fs/vx_ioctl.h>`, which contains the following members:

```
ino_t            ext_inum;      /* inode number */
long             ext_igen;      /* generation count of file */
off_t            ext_isize;     /* size of file in blocks */
off_t            ext_iblocks;   /* blocks held by file */
off_t            ext_offset;    /* offset in file to reallocate from */
off_t            ext_size;      /* number of blocks to reallocate */
struct extalloc  ext_alloc[5];  /* type of reallocation to do */
struct dinode    *ext_inode;    /* pointer to inode */
```

Attempt to reallocate extents in the file specified by `ext_inum`. The `ext_igen`, `ext_isize`, and `ext_iblocks` fields are used to validate the inode. If these fields are nonzero, and they do not match the values in the inode, then the reorganization will fail.

The `ext_offset` and `ext_size` fields define an area of the file. The `ioctl` will attempt to allocate new extents for that area. If successful, then the file data will be copied to the new extents, and the old

extents for that area will be released. This can be used to move files to a different location, or to combine small extents to reduce fragmentation.

The `ext_alloc` array of `extalloc` structures determines how the reallocation will occur. The `extalloc` structure is defined in `<sys/fs/vx_ioctl.h>`; it contains the following members:

```
long      ext_cmd;       /* type of reallocation */
long      ext_flag;      /* flag indicating search direction */
daddr_t   ext_start;     /* starting block for search */
daddr_t   ext_end;       /* ending block for search */
long      ext_nextents;  /* max number of extents to allocate */
```

The `ext_cmd` field determines the type of search to make. The possible values are:

`EXT_SEARCH`
: Search the area from `ext_start` to `ext_end` for one extent of `ext_size` blocks. The `ext_flag` field controls the direction of the search. If `ext_flag` is `EXT_FORWARD`, then the search runs from `ext_start` to `ext_end`. If the `ext_flag` is `EXT_BACKWARD`, then the search runs from `ext_end` to `ext_start`. If a search hits the end of the file system, then it wraps around to the other end.

`EXT_PREALLOC`
: Use the kernel's space preallocation algorithm to allocate extents totaling `ext_size` blocks. If more than `ext_nextents` are required, then fail the attempt.

Any extents allocated by the reallocation must meet all the files restrictions on extent size, alignment, and contiguity.

The `ext_inode` field is a pointer to a disk inode structure. If it is not NULL, then on a successful reallocation, the new inode will be copied into the specified `dinode` structure, as defined in `<sys/fs/vx_inode.h>`.

This command may only be used by a user with appropriate privilege on the root directory of the file system.

The `VX_EXTREORG ioctl` returns a 0 if all of the reallocation attempts failed. If a reallocation occurred, then the return value is 1 plus the index of the successful reallocation command in the `ext_alloc` array. If the operation fails, the return value is −1 and the external variable `errno` will be a general diagnostic or the following:

`EINVAL`
: The file specified by *fildes* is not a regular file. The file specified by *fildes* has allocated indirect extents. The file size, block allocation, or generation count does not match that specified in the `extreorg` structure.

DIAGNOSTICS

The following values are returned in `errno` upon operation failure:

`EPERM`
: The process does not have appropriate privilege.

`ENODEV`
: The file specified by *fildes* is not the root directory of an Veritas file system.

`EROFS`
: The file system is mounted read-only.

`EIO` An I/O error occurred while attempting to perform the operation.

`ENOSPC`
: Requested space could not be obtained.

`EFAULT`
: An address specified by an argument is invalid.

SEE ALSO

ioctl(2), getrlimit(2), ulimit(2).

System Administrator's Reference Manual

zero(7)

NAME

`zero` - source of zeroes

DESCRIPTION

A zero special file is a source of zeroed unnamed memory.

Reads from a zero special file always return a buffer full of zeroes. The file is of infinite length.

Writes to a zero special file are always successful, but the data written is ignored.

Mapping a zero special file creates a zero-initialized unnamed memory object of a length equal to the length of the mapping and rounded up to the nearest page size as returned by `sysconf`. Multiple processes can share such a zero special file object provided a common ancestor mapped the object `MAP_SHARED`.

FILES

`/dev/zero`

SEE ALSO

`fork`(2), `mmap`(2), `sysconf`(3C).

System Maintenance Procedures(8)

Introduction

System Administrator's Reference Manual

intro_sarm8(8)

NAME

`intro_sarm8` - introduction to system maintenance procedures

DESCRIPTION

This section outlines certain procedures that are of interest to those responsible for system maintenance. Included are discussions of such topics as boot procedures, recovery from crashes, file backups, etc.

SEE ALSO

System Administrator's Guide.

autoconf(8) - ldcontroller(8)

System Administrator's Reference Manual

autoconf(8)

NAME

`autoconf` - display hardware configuration

SYNOPSIS

`autoconf –l [–v]`

`autoconf –p`

`autoconf –a [–v]`

`autoconf –d` *devname*

`autoconf –f`

```
autoconf –F
```

DESCRIPTION

`autoconf` displays information about the various hardware components of the system and the current state of the associated software.

Hardware configuration information is maintained by the kernel in a set of data structures resembling a tree. Each data structure or node represents a single hardware device. The nodes are connected together in a way that usefully represents the way the hardware is connected together. Controllers are attached to buses, and devices are connected to controllers.

OPTIONS

`–l` Shows the hardware configuration. Each line describes a node in the kernel `autoconf` tree. Indentation is used to indicate where in the tree the node appears. The root node is leftmost. The branch and leaf nodes are displayed progressively to the right. Each line contains the device name and a descriptive string from the node.

`–v` The verbose, `–v`, flag adds node type and state information to the beginning of each line. The state information is described by several letters. The letter `F` indicates that the device was found by the software. `E` indicates that the device has been enabled for use. `B` indicates that the device is broken and may not work reliably if still enabled.

The following is an example of the output lines that can occur when calling `autoconf –lv`:

```
DEV:FE  O :D  RX :              ios0/sdisk040 MP14 -- MP14
```

"O" stands for "open", which means that one of the disk partitions is currently in use. If there is an "S" (suspended) here instead of an "O", the device cannot be used at this time.

`–p` Shows the hardware configuration in an easily parsable format. This option is equivalent to the `–l` option except that fields in each line of output are separated by "|" characters.

`–a` Shows all information about the `autoconf` node, such as tree connectivity, memory requirements, and any hardware specific information (i.e. SCSI inquiry name). This format is primarily useful for field engineers.

`–d` *devname*
Outputs the major number and minor number of the device specified in *devname*. The device name must be specified as follows, for example: `ios0/sdisk000s0` or `ios0/stape000h`.

`–f` Shows the inquiry data (device name + firmware revisions) of all SCSI devices (disks, tape drives etc.).

`–F` Shows the `autoconf` name, the universally unique name for this FC (Fiber Channel) connection and the current FC loop Id for all FC devices (disks, RAID systems etc.).

DIAGNOSTICS

Exit status is normally `0`. If an error occurs, exit status is `1`.

NOTES

Some device drivers may not use the `autoconf` kernel facilities. `autoconf` will not produce any output for these devices.

FILES

```
/dev/autoconf
```

SEE ALSO

`devname(7)`, `sdisk(7)`, `sraid(7)`.

System Administrator's Reference Manual

boot2(8)
RM600 only

NAME

`boot2` - program used to boot Reliant UNIX on an RM600

DESCRIPTION

`boot2` can be loaded with `boot1` from tape, disk or CD-ROM, and is used to bootstrap Reliant UNIX.

`boot2` loads the Reliant UNIX operating system on to RM600 systems. It is loaded into the main memory using the `boot1` command from a particular hard disk, tape, or CD-ROM. The commands are then processed by `boot2` to load the operating system and then pass control to it when the load operation is completed.

SCSI I/O disk, tape, and CD-ROM devices can be used as boot devices. Furthermore, `boot2` can use any disk drive with a `ufs` file system to load any kernel executable file on that device. A previous mount of this device is not necessary.

Tape devices or a CD-ROM are generally used to install the operating system software. It is possible, however, to boot only a kernel from a tape device or a CD-ROM. This is useful for system recovery when the boot disk has been corrupted, preventing `boot2` from reading the disk.

The `boot2` command parser outputs the prompt `boot2:`, while `boot1` on the other hand gives the prompt `BDM(MBII 0 t)>`. The `boot2` commands are listed with a description of their syntax and a description of all optional and mandatory arguments.

By pressing CTRL-A (on an RM600) or CTRL-@ BDM (on an RM600-E), `boot2` is forced to return control to the RM600 firmware. It can then be loaded again from that point.

The boot2 commands

The following list contains the commands and options provided by `boot2`, which can be entered at the prompt `boot2:`. The commands are listed in alphabetical order and the boot variables named are described under `bootvariable`(7).

- `boot` [–n] [–f *file*] [*args*]

  Loads a file from the specified device and executes it.

  –n For Siemens Nixdorf internal development only. The operating system executable file is loaded but not started. After `boot2` has loaded the executable file, `boot2` returns to the command parser. A subsequent boot command must be used to load and/or start another executable file. This command is useful when an executable file is loaded in addition to a debug monitor. An executable file loaded with this option can be started later with the `go` command.

  –f *file*

  Specifies the name of the file to be loaded. With operating system version V5.43 or later, the name can be specified logically or physically. It is only possible to specify disks as root devices using the logical name. The root disk specified using a logical device name must be entered in the root partition list in CSI/HIOS-NVRAM (this is done when the disk is installed). The logical root disk name conforms to UNIX disk name notation [see `sdisk`(7)].

  Example: `sdisk022s0`

  If the root device name is specified physically (conforms to the old notation), the filename can have one of the following formats:

  Tape:

  `st`([*host_adapter_no.*],[*SCSI_ID**8],)*file_name*

  If the device name `st` is specified, the tape is rewound before and after it is read. Specifying an `n` before the device name (`nst`) suppresses the rewind before the read, and an `n` after the device name (`stn`) suppresses the rewind after the read.

Disk:

`sd`([*host_adapter_no.*],[*SCSI_ID**8],[*partition_no.*])*file_name*

CD-ROM:

`cd`([*host_adapter_no.*],[*SCSI_ID**8],)*file_name*

If one of the optional arguments is not specified, its value is set to `0`.

args Arguments which are to be passed to the booted image when immediately started.

Examples:

```
boot2: boot initarg=-s
```

The `/unix` file is loaded from the disk labeled active in the root partition list and executed until the operating "S" has been reached:

```
boot2: boot –f /unix.old
```

The `/unix.old` file is booted from the disk labeled active in the root partition list.

- `cp` [`–c` *count*] *file1 file2*

  Copies the contents of *file1* to *file2*.

  file1 is the file to be copied, and *file2* is the file to which it is to be copied. *file1* and *file2* can be raw devices.

  While the command is being executed, a period (.) is printed on the screen for each record transferred.

  `–c` *count*

  Specifies the maximum byte count to be transferred. If `–c` is not specified, the copy terminates at the EOF character of *file1*.

  Example:

  To copy *file3* (the fourth file; *file0* is the first file) on the streamer tape to partition 1 of the disk, type:

  ```
  boot2: cp st(0,32,3) sd(0,0,1)
  ```

- `go` [*address*]

  This starts an executable file previously loaded into memory with the `boot2` command.

  address

  Specifies the address where the execution of the previously loaded image is to be started. This is useful when there are several images loaded and one address is to be selected. If this argument is omitted, the image most recently loaded is started.

- `help` [*commandlist*]
- `?` [*commandlist*]

  Displays the syntax for all commands in *commandlist*.

  commandlist is one or more commands separated with a space.

  If you do not specify *commandlist*, then the `help` command displays the syntax for all commands. You can also enter a question mark (?) instead of the command name `help`.

- `loadenv`

  Loads the boot variables stored in `/etc/default/boot` as the current boot variables. These variables can then be displayed using `printenv`, or they can be modified with `setenv` or `unsetenv`. The current boot variables are transferred using the `boot` command.

- `nv_backup`

  Calculates the checksum via the software parameter area of CSI/HIOS-NVRAM and stores the software parameter area (including the resultant checksum) on the system cabinet disks.

  Example: `boot2: nv_backup`

- `nv_list`

  Lists the checksums of the CSI/HIOS-NVRAM backups stored on the system cabinet disks.

  Example: `boot2: nv_list`

- `nv_restore` *phys_part*

  Selects the CSI/HIOS-NVRAM backup for the specified disk and copies the data to CSI/HIOS-NVRAM.

  Example: `boot2: nv_restore sd(1,32,0)`

- `printenv` [*varlist*]

  Displays the values of the boot variables in *varlist*.

  varlist

  Is a special boot variable, or several variables separated by blanks. If no boot variables were specified, `printenv` prints all the defined boot variables.

  Example:

  The following example prints the value of the `bootfile` boot variable.

  ```
  boot2: printenv bootfile
  bootfile = sd(0,0,0)unix
  ```

- `root`

  With operating system version V5.43 or later, root disks are installed in the system in logical boot mode by default. In this mode, up to 10 root disks can be administered in a system. The root disks can be specified logically or physically. When a root disk is installed in the system, an entry is made in the root partition list for the disk to be installed. During normal operation, these entries are administered automatically. The `boot2` command enables entries in the root partition list to be corrected (e.g. repair of incorrect entries, integration of a root disk clone).

- `root –l`

  Outputs the root partition list for CSI/HIOS-NVRAM.

  Example: `boot2: root –l`

- `root –n` *entry* `–l` *log_part*

  Writes the partition name specified with *log_part* to the entry in the CSI/HIOS-NVRAM root partition table defined using *entry*. If *log_part* is not specified, the entry is deleted from the root partition table.

  Example: `boot2: root –n 1 –l sdisk010s0`

- `root –n` *entry* `–w` *swap_part*

  Writes the partition name specified with *swap_part* to the entry in the CSI/HIOS-NVRAM root partition table defined using *entry*. If *swap_part* is not specified, the entry is deleted from the root partition table.

  Example: `root –n 4 –w sdisk010s1`

- `root –n` *entry* `–p` *phys_part*

  Writes the partition name specified with *phys_part* to the entry in the CSI/HIOS-NVRAM root partition table defined using *entry*. The disk ID of the disk specified using *phys_part* is also entered in *entry*. If *phys_part* is not specified, the entry is deleted from the root partition table.

  Example: `root –n 0 –p sd(0,16,0)`

- `root –n` *entry* `–s` *status*

  Writes the partition name specified with *status* to the entry in the CSI/HIOS-NVRAM root partition table defined using *entry*. If *status* is not specified, the entry is deleted from the root partition table.

  The status values are defined as follows:

  ```
  ROOT_UNDEFINED
       0x0000
  ROOT_INACTIVE
       0x0001
  ROOT_ACTIVE
       0x0002
  ```

  Example: `root –n 3 –s 2`

- `root –n` *entry* `–d`

  Deletes the root partition table entry specified using *entry*.

  Example: `boot2: root –n 4 –d`

- `setenv` [*var*] [*value*]

  Creates a new boot variable or changes the value of an existing boot variable.

  var Is the boot variable you want to set.

  value
  Is the value to which you want to set the variable.

  Boot variables are represented as ASCII strings. The current values of these variables are passed to the programs booted by `boot2`.

  Example:

  The following example defines the boot variable `bootfile`.

  ```
  boot2: bootfile = sd(0,0,8)unix
  ```

- `unsetenv` [*var*]

  Deletes a boot variable.

  var Is the variable to be deleted.

  Example:

  The following example deletes the definition of the boot variable `bootfile`.

  ```
  boot2: unsetenv bootfile
  ```

SEE ALSO

`bootvariable(7)`.

System Administrator's Reference Manual

bootdisk(8)
RM600 only

NAME

`bootdisk` - mark a root disk as the next activeboot disk

SYNOPSIS

`bootdisk` *devicename*

DESCRIPTION

`bootdisk` is a command for marking a boot disk which can be used as the active root disk for the next reboot. The root disk is selected using *devicename*.

The *devicename* parameter must be specified as a block-oriented device name, whereby the partition must not be defined [see `sdisk(7)`].

A number of commands are executed in the system in order to mark and activate the root disk as the next boot disk:

1. The disk ID of the disk *devicename* is determined.
2. The logical device name is entered as a bootstring using the `bootflags(8)` command.
3. The disk is entered in the root partition list as the active root disk and overwrites the previously active entry. The root partition list is a list of data that is activated using the boot program `boot2(8)` .

DIAGNOSTICS

`0` `bootdisk` executed the necessary commands successfully.

`≠ 0` An error has occurred. `bootdisk` outputs an error message.

EXAMPLE

Your root disk has, e.g. just failed or is defective, and the system continues to run on the mirrored root disk (`/dev/ios0/sdisk009s0` in the example). The disk `/dev/ios0/sdisk009s0` must now be labeled as the active root disk so that the system can be booted from it the next time a reboot takes place. You do this by issuing the following command:

```
$ bootdisk sdisk009
```

WARNING

The `bootdisk` command should only be used for selecting the active root disk from the mirror root disks.

If the command is used to activate other root disks (in particular for those originating from another system), serious data loss may occur due to the use of configuration files that are not consistent with the system hardware.

SEE ALSO

`sdisk(7)` , `boot2(8)` , `bootflags(8)` .

System Administrator's Reference Manual

bootflags(8)
RM600 only

NAME

`bootflags -` display/modify operating system boot flags or modemoperating mode

SYNOPSIS

`bootflags` [`–p`] [`–f` *bootflags*] [`–b`*bootstring*]

`bootflags` [`–s` | `–t` | `–c`]

`bootflags –d` [*dialstring*]

`bootflags` [`–R` *val*] [`–D` *val*]

DESCRIPTION

`bootflags` allows you to display and modify values which control the booting of the operating system. These values include the boot flags and the boot string. They can be set for permanent or temporary use. Permanent values are even valid after powering down the machine.

`bootflags` also can be used to switch between the two modem operating modes "Teleservice" and "Remote Console", to display the current state and to display or modify the dial string for the "Remote Console" modem connection.

If you call `bootflags` without specifying any options, the current boot flags and the current boot string are displayed.

OPTIONS

`–p` Displays the permanent values for bootflags and bootstrings. In combination with the `–f` or `–b` options, the permanent values for bootflags and bootstrings are set.

`–f` Sets the *bootflags* specified. The boot flags are described below.

`–b` Sets the *bootstring* specified.

`–s` Displays the current modem operating mode, the modem permissions and the connection state.

`–t` Switches from Remote-Console mode to Teleservice mode.

`–c` Switches from Teleservice mode to Remote-Console mode.

`–d` Sets the *dialstring* specified. Without a dialstring specified displays the current dial string for Remote Console.

`–R` Sets the Remote Console permission to the specified value *val*.

`–D` Sets the dial in permission to the specified value *val*.

val The following values *val* are permitted:

`0` permission denied

`1` permission allowed

Bootflags

The following values are permitted for the *bootflags*:

`0x0000`
System autoboot *)

`0x0001`
Manual Reliant UNIX boot

`0x0008`
Stop reboot at monitor

`0x0010`
Don't reboot, start BDM

`0x0020`
Power off system

`0x0040`
Stop BBU

`0x0200`
Automatic boot. Default value after installation.

The values can be combined (e.g. `0x0009` to manually boot the entire system), but please note that not all possible combinations are practical.

*) This boot option is overridden when the system is started by `init(1M)` with option `5`.

Bootstring

With operating system version V5.43 or later, the bootstring can be specified logically or physically.

The logical bootstring conforms to UNIX disk name notation [see `sdisk(7)`].

Example: `sdisk022s0`

The physical bootstring is structured as follows:

device([*channel_id*],[*scsi_id*`*8`],[*partition*])

The parameters have the following meaning:

device
Name of the boot device, where the following can be specified:

`cd` Boot from CD-ROM

`sd` Boot from hard disk

`st` Boot from tape

channel_id
This parameter specifies the SCSI bus where the boot device is situated.

scsi_id
Number of the device on the SCSI bus. This number must be multiplied by the factor `8`.

partition
Partition from which Reliant UNIX is to be booted. This parameter is only evaluated by `boot2` when Reliant UNIX is booted, if the variable `BOOTFILE` is not set and the Reliant UNIX boot string is not overridden.

DIAGNOSTICS

`bootflags` returns exit code 0 if no error occurs; otherwise it prints a diagnostics and returns non-zero.

Exit codes:

`1` = Syntax error.

`2` = Open on file `/dev/sm_cons` failed.

`3` = Read of values failed.

`4` = Set of values failed.

For detailed information on error situations refer to `sm_cons(7)` .

FILES

`/dev/sm_cons`

SEE ALSO

`bootvariable(7)` , `sm_cons(7)` , `boot2(8)` , `showconf(8)` .

System Administrator's Reference Manual

changedev(8)

NAME

changedev - physically change the attached devices

SYNOPSIS

changedev [–msN] –a *port*

changedev [–msN] –d {*device* | *port*}

changedev [–mpN] –D {*device* | *port*}

changedev –x

Options for OLR/LAR/HV scripts only (–s is set implicitly):

changedev [–mN] –f {*device* | *port* | *controller*}

changedev –F *device*

changedev [–mN] –g {*device* | *port* | *controller*}

changedev [–N] –S {*device* | *port*}

changedev [–N] –u {*device* | *port*}

changedev [–mN] {–0 | –1} {*device* | *port*}

DESCRIPTION

changedev is the default command for OLR (Online Replacement) and the OBSERVE high-availability application. It can be used to add, remove, or shut down devices on a system that is running. Use CONFIG to replace a device while the system is running.

Unlike the reinitdev(8) command, the underlying hardware (device, bus, controller) is not initialized (reset).

device is the SCSI device name as it is displayed with autoconf(8) . Example: ios0/sdisk033.

port is the SCSI bus or port name as it is displayed with autoconf(8) . Example: ios0/sport05.

controller is the SCSI controller name as it is displayed with autoconf(8) . Example: ios0/scon07.

OPTIONS

–a Adds one or more devices to the specified SCSI port. The SCSI port must be specified using its name as it is displayed with autoconf(8) . The SCSI port must have space for the device to be added. The address setting on the new device must not conflict with the address of any existing devices on that port. Use the autoconf –l command to examine the current state of the SCSI subsystem in order to determine where a device may be added.

changedev verifies that space is available on the specified port and prompts the user to insert the new device. When the insertion is finished, changedev probes that port and initializes any new devices that are found. The user is given an opportunity to review the results and make additional changes as needed before exiting the utility.

The command does not affect any devices already present.

changedev has no knowledge of how many drives the user intends to add. The user must verify that the additions are correct.

–d *device*

–d *port*

This option is used in an OBSERVE environment and removes the kernel structures of the affected devices. The SCSI bus is then searched for connected devices and any devices found are reconfigured back into the kernel.

changedev –d generally works interactively to allow the user the opportunity to remove disks.

If the –s option is also specified when using the –d option, the relevant SCSI bus is not searched for connected devices, as was the case with earlier versions of changedev.

–D *device*

–D *port*

Removes all associated kernel structures of a specified device not currently being used. A subsequent autoconf –l call is not displayed and the device can no longer be accessed. In contrast to the –d option, connected devices are no longer sought, after the bus has been deconfigured.

If a port is specified as a parameter and if all connected devices are not in use at present, the changedev –D *device* functionality will be executed for each of these devices.

changedev –D does not work interactively and does not produce any output. If the specified device is opened (or if at least one device is opened when a port is specified) the changedev –D *device* functionality is not executed for any one device. Since in this case, the autoconf status of the connected devices is re-initialized, any changedev –f executed beforehand for this device or port is canceled again. This means that these devices can then be accessed again.

You have to evaluate the exit status (0 = ok, 1 = device reserved) to detect whether an error has occurred.

The –D option is a shortcut, as it were, for –s –d.

–N If this option is used, the relevant system files for CONFIG and LAR (e.g. /etc/device.tab) will not be updated, as a result of which the runtime is drastically reduced. However, this option should only be used by experienced system administrators, since they are the ones ultimately responsible for the consistency of these files.

–m This option is set implicitly for all options except –a.

The –m option must be used with the –a, –D, –f, or –g options if *device* or devices on *port* are multi-host devices (devices shared by nodes in a cluster).

The –m option reads from the /etc/clustertab configuration file by default to obtain hosts in the multi-host cluster. If the OLMCONF environment variable is set, changedev uses the file given in the OLMCONF variable as the configuration file. For information on the format of the configuration file, see dlmconfig(8) .

The –m option also requires the nodes in the cluster to be set up as trusted hosts for online maintenance (OLM). Create an .rhosts file under the /home/olmadmin directory that contains the names of all nodes in the cluster. Put one name on each line. Before beginning any actions, the –m option prompts the user for a list of hosts that are up and running. A message is then printed to the terminal screen confirming the list of live hosts, and the user is given another opportunity to change the list.

With the –a option, changedev verifies that space is available on the specified port and prompts the user to insert the new device. When the insertion is finished, changedev executes a command on the live hosts to initialize any new devices that are found.

–s The –s option is used to do the work silently, i.e. the user is not prompted for things like "Do you want to continue?". Also much of the normal output is not printed.

–p (only in conjunction with –D) If the device is a disk, its motor is switched off.

–f *device*

This option is used by LAR, for example, to prevent a defective device being accessed. This can be shown with autoconf –lv:

Beforehand:

```
DEV:FE    :D  RX :              ios0/sdisk022 MP36 -- MP36
```

After changedev –f is called:

```
DEV:F     :D  RX :              ios0/sdisk022 MP36 -- MP36
```

This means that the ENABLE flag is reset for this device.

The following situations can be distinguished here:

a) *device* not currently in use (`autoconf -lv: DEV:FE :D RX : ...`):
Device is locked to prevent future access.
Exit status = 0.

b) *device* currently in use (`autoconf -lv: DEV:FE O :D RX : ...`):
No action.
Exit status = 1.

c) *device* does not exist:
No action.
Exit status = 1.

`-f` *port*
`-f` *controller*

All connected devices are locked to prevent future access. Current access is not affected by this.

Warning: A subsequent `changedev -D` issued for this port clears the lock again!

If the specified port or controller does not exist, the exit status = 1, otherwise the exit status is always 0, even if devices are currently opened!

An example of this: Beforehand:

```
BUS:FE    :    X :          ios0/sport03 FAST8SE activ Id: 7
DEV:FE    :D  RX :            ios0/sdisk030 MP12 -- MP12
DEV:FE  O :D  RX :            ios0/sdisk035 MP12 -- MP12
```

After `changedev -f` is called:

```
BUS:F     :    X :          ios0/sport03 FAST8SE activ Id: 7
DEV:F     :D  RX :            ios0/sdisk030 MP12 -- MP12
DEV:F   O :D  RX :            ios0/sdisk035 MP12 -- MP12
```

`-F` *device*

This option is used by CONFIG to exchange a device with replaceable media, such as a magnetic tape device. *device* specifies the device to be replaced. `changedev` ensures that

- an `open` is not active on any device with replaceable media, which is connected to the same port as *device*;
- an inserted medium is at loadpoint for all magnetic tape devices connected to the same port as *device*;
- adevice is not inserted in the device being replaced.

This check only takes account of devices that can still be accessed at the time they are invoked, with the exception of the device to be replaced. All devices for which the above conditions apply are then locked to prevent further access.

To check whether this was possible for all devices, the output of `changedev` can be interpreted or the exit status evaluated. The following output line appears for any device that could not be locked:

rawdevicename x

where *x* can be a number between 1 and 3 and have the following meaning:

x=1 an `open` is still active on the device specified by *rawdevicename*;

x=2 amedium has been inserted in the device specified by *rawdevicename* which is not at loadpoint;

x=3 amedium is inserted in the device being replaced.

The exit status indicates the number of devices which could not yet be locked for the above reasons. The exit status is 0 if all devices fulfill the conditions given above and can be locked to prevent further access, i.e. the device can now be exchanged.

Diskette drives cannot be locked and the device is also not checked to see if a medium is inserted. The replacement of a device (OLR) is not supported in all systems. Please refer to the Operating Manual for

your system for further information.

Once a device has been replaced, the access lock can be lifted again by calling `changedev -g` for all devices connected to a port or for a single device.

`-g` *device*
`-g` *port*
`-g` *controller*
: The access lock is cleared for all affected devices. The `-g` option therefore has the opposite effect to `-f`.

Exit status = 0 if the device/port/controller exists, otherwise = 1.

`-S` *device*
`-S` *port*
: Suspends the specified device from a remote host. The `-S` option can only be executed by a `changedev -m` command on a remote host.

`-u` *device*
`-u` *port*
: Powers up the specified device from a remote host. The `-u` option can only be executed by a `changedev -m` command on a remote host.

`-0`
`-1`
: Specifying `-0` deactivates all mirror disk pieces located on affected physical disks; specifying `-1` activates all these mirror disk pieces and starts the catch-up process again [see `dkmirror(8)`, `-d` and `-e` options].

`-x` Displays the option usage.

NOTES

It is possible to remove more than one device at a time but it is not recommended. When one device is removed at a time, it is much simpler to verify that the correct device has been specified and removed. When more than one device is removed, a warning is sent to the terminal screen. The user can choose to make a correction or continue. This warning is also issued if a user removes a device other than the device specified.

`changedev` must not be used for devices containing the root file system or active swap partitions.

Parallel activity by `reinitdev`/`changedev` is only permitted using different SCSI buses.

SEE ALSO

`gettypes(1)`, `hosts.equiv(4)`, `devname(7)`, `autoconf(8)`, `reinitdev(8)`.

System Administrator's Reference Manual

clean8mm(8)

NAME

`clean8mm` - preparations for cleaning an 8mm magnetic tape drive

SYNOPSIS

`clean8mm` [`–q`] [*tapename*]

DESCRIPTION

`clean8mm` prepares an 8mm magnetic tape drive for cleaning. Once `clean8mm` has been called, the device can be cleaned in accordance with the instructions on the cleaning cassette.

If you have already inserted the cleaning cassette before calling `clean8mm`, the cassette must be inserted again after you have called up `clean8mm` (close or push in door).

The `–q` option suppresses messages from `clean8mm`. *tapename* must correspond to the format described under `stape(7)` . If no *tapename* is specified, the environment variable `TAPE` must exist and contain *tapename*.

NOTE

`clean8mm` must be called as the "Auto Load" option on the 8mm magnetic tape drive is normally switched off. Cleaning can only take place, however, when the "Auto Load" option is active. After `clean8mm` has been called, the "Auto Load" option remains switched on until the next time the tape is accessed normally.

FILES

`/dev/ios0/rstape*`

SEE ALSO

`devname(7)`, `mtio(7)`, `stape(7)`, `autoconf(8)`.

Instructions on the cleaning cassette.

System Administrator's Reference Manual

cleandev(8)

NAME

`cleandev` - check whether tape drives need to be cleaned

SYNOPSIS

`cleandev`

DESCRIPTION

`cleandev` checks whether the cleaning bit is set on tape drives that support a cleaning bit. If the cleaning bit is set, a message is output to the console, and with the help of the LOG3 logging systems, the message is also written to a log file.

The console message is also output after each job for the relevant tape drive.

If a message is displayed, the user must make sure that the drive is cleaned.

Note: With some drives the cleaning bit is also set for other events (e.g. if the number of load processes is exceeded for the tape medium).

With the help of `sysadm(1M)` , the log file can be searched for this special entry.

If the cleaning bit is not set, nothing happens.

NOTES

If the cleaning bit is set, the following message appears on the console:

.IP "`sti:781: It's time to clean the drive iosX/stapeXXX`".

This message is output by the operating system every time `cleandev` is invoked and after every job for the tape drive where the cleaning bit is set.

When the `cleandev` command is issued, the following message is output in addition to the console message on the terminal from where `cleandev` was called:

"`It's time to clean the drive iosX/stapeXXX`"

Along with the message text, the following entries are simultaneously entered into the log file:

componentno.
: 1080008

structid
: 70

level
: WARNING

errorno.
: 100

The cleaning bit is supported by all 8mm EXABYTE, 4mm DAT, and by the PXT1-B1 (A or 11) drives. The cleaning bit cannot be evaluated with any other drives.

Please also note the following in relation to 8mm EXABYTE tape drives (with the exception of MC75):

The cleaning bit is not supported for all types of tape drive and therefore, cannot be evaluated in all cases.

You generally clean the device once you have been requested to do so, and then reinsert the tape. If the cleaning message appears again immediately (either via the `cron` routine or when you start the `cleandev` command again), it may be because the tape was already loaded more than 500 times. Poor operating conditions (e.g. dusty environment, start/stop mode) will reduce the cartridge lifetime to 30 operating cycles. Once you have saved the relevant data, you should no longer use the tape. See also the leaflet "How to use 8mm Data Recorders", Order No.: 86329.00.1.93

FILES

`/etc/conf/pack.d/sti/space.c`
: Configuration file for tape driver

`/var/spool/cron/crontabs/root`

`/opt/log3/default`X`log`
: Usually the LOG3 default log file (X = 0 – 4)

SEE ALSO

`stape`(7), `clean8mm`(8).

System Administrator's Reference Manual

del_dbg_kernel(8)
RM600 only

NAME

`del_dbg_kernel -` delete debug kernel environment

SYNOPSIS

```
/usr/sbin/del_dbg_kernel –a
/usr/sbin/del_dbg_kernel –c
/usr/sbin/del_dbg_kernel –h
```

DESCRIPTION

This command deletes the packages of the debug kernel, whose names have been stored in an information file using `inst_dbg_kernel(8)` . In addition, the directories used for the debug kernel are deleted under `/.utsdbg/etc/conf` in the directory tree.

Please refer to the `utsdbg(5)` manual page for an overview of the installation procedure, generation, as well as the activation and deletion of the debug kernel.

OPTIONS

`–a` The debug kernel environment is deleted without any queries by specifying the `–a` (armed mode) option.

The packages of the debug kernel are deleted as well as directories relating to the debug kernel under `/.utsdbg/etc/conf`.

`–c` A number of conditions required for the successful execution of `del_dbg_kernel -a` are checked by specifying the `–c` (check mode) option. No changes are made to the debug kernel environment, and in particular no deletion is carried out with `–c`.

If `del_dbg_kernel -c\f returns with a value other than 0, it will also not be possible for del_dbg_kernel -a` to be executed successfully.

`–h` An explanatory syntax message is displayed.

DIAGNOSTICS

Since `del_dbg_kernel` invokes other commands, such as `pkgrm(1M)` for example, in order to delete the environment for generating the debug kernel, errors detected with this command are also reported. In general terms, the return code 1 or 10 indicates that `del_dbg_kernel` has detected an error.

The return code 10 indicates a syntax error when invoking `del_dbg_kernel`.

The return code 1 is used for other errors that have occurred.

All error messages and warnings should be comprehensible without further explanation.

FILES

The package names installed with `inst_dbg_kernel(8)` (value of the `PKG` package variable) are recorded in a file and used for information purposes when deleting with `del_dbg_kernel`.

SEE ALSO

`pkgrm(1M)` , `utsdbg(5)` , `inst_dbg_kernel(8)` , `setup_dbg_kernel(8)` .

System Administrator's Reference Manual

dkconfig(8)

NAME

`dkconfig` - virtual disk configuration utility

SYNOPSIS

`/sbin/dkconfig [ -clinNqsuvwW ]` *special-device ...*

`/sbin/dkconfig [ -clinNqsuvwW ] -a|-A|-g`*N*`|-O`

`/sbin/dkconfig -R[ vn ]` # for Mirror Root/Swap

`/sbin/dkconfig -zlv`

`/sbin/dkconfig [ -tp ]` *special-device ...*

`/sbin/dkconfig -k [ -v ]`

`/sbin/dkconfig {-C[ v ]|-L[ v ]}` # for Mirror Root/Swap

`/sbin/dkconfig [ -x ]`

DESCRIPTION

The `dkconfig` command allows the configuration to be controlled from virtual disks (vdisks); the definitions of the configuration files for Vdisk Lite [VL, see `vl(7)`] and `/etc/dktab` are used as a basis here.

`dkconfig` reads `/etc/dktab` to obtain virtual disk configuration information in the same way that `mount` reads `/etc/vfstab` to obtain mountable file system information. `dkconfig` configures the virtual disks using the definitions in the VL configuration files and in `/etc/dktab` and also creates an overview of the current configuration of the virtual disks. As such, it is important that definitions of configured virtual disks in `/etc/dktab` are not altered so long as those virtual disks remain configured and in use. However, definitions of unconfigured virtual disks may be altered at any time. A detailed description of the format and use of this disk configuration file is found in `vl(7)` and `dktab(4)` .

When multiple virtual disks are configured, `dkconfig` checks for any accidental physical disk overlap in the set of virtual disks being configured. Nevertheless, when defining a virtual disk, the administrator should always check that its segments do not overlap already existing file system partitions, data or other already-configured virtual disks. The administrator must exercise great care when defining this type of virtual disk in order to avoid using it to overwrite and therefore corrupt a file system or data that already exists.

On the other hand, there is no risk involved in simply configuring a virtual disk regardless of its definition. It is only once the virtual disk is put to use and data is actually written to it, that there is a risk of endangering current file systems or data. In fact, virtual disks may be configured and unconfigured at any time. However, once configured and in use, a virtual disk cannot be unconfigured. An `EBUSY` error is returned if any attempt is made to unconfigure a busy virtual disk.

If any virtual disks defined in the configuration files contain file systems, `dkconfig` must be executed prior to `fsck` or `mount` because a virtual disk must be configured before it can be used. This actually happens automatically as part of the process of going to multiuser mode; the script `/sbin/brc` calls up `dkconfig` to configure the virtual disks in accordance with `/dktab`. This does *not* apply, however, for grouped virtual disks. In order to ensure that grouped virtual disks are also configured when switching to multiuser mode, you should either enter "`/sbin/dkconfig -vcg`*N*" as the last line in `/sbin/brc` for each group, or alternatively modify a script in `/etc/rc2.d`.

To review file systems on virtual disks manually in single-user mode, you must execute `/sbin/brc` from the command line before you call `fsck`. Failure to configure a virtual disk before trying to run an `fsck` on it causes an `ENXIO` (No such device or address) error to be returned.

Although all virtual disk pathnames specified in `/etc/dktab` or as arguments to `dkconfig` are block special nodes, both the block and the character special nodes are supported. For a detailed description of virtual disks, refer to the `vdisk(7)` manual page.

OPTIONS

`-A` Configures all disks defined in `/etc/dktab`.

`-a` Invokes operation for all VL vdisk groups created with the `-L 0` option [see `vdadmin(8)`] as well as for all non-grouped disks (default) defined in `/etc/dktab`.

`-C` Displays the current `mroot` configuration (mirror root disk and mirror first swap area) in accordance with `mrconfig.*`.

`-c` Configures virtual disks.

`-g`*N* Configures all virtual disks defined in `/etc/dktab` in group *N*.

`-i` Uses `stdin` instead of the `/etc/dktab` file; The system database for VL is always used.

`-k` Lists virtual disks currently configured in the kernel.

`-L` Displays the current `mroot` configuration (mirror root disk and mirror first swap area) in accordance with `/etc/dktab`.

`-l` Lists current virtual disk configuration.

`-lv` Lists I/O statistics for each segment of a virtual disk.

`-N` Bypasses boundary checking when the virtual disk is configured; this ensures that physical segments are not opened during configuration. Do not use this option unless the virtual disk parameters are legal (for example, you are sure the boundaries are not overlapping) (only together with the `-c` option).

`-n` (Un)configures mirror disks while simultaneously deleting the statuses of the individual segments of the mirror disks (only together with the `-c`, `-u`, or `-R` option).

The `-n` option should be used if a mirror disk is being configured for the first time or if your configuration was modified in `/etc/dktab` (e.g. if the number of pieces you have has been reduced).

`-O` Configures all OPS disks defined in `/etc/dktab`.

`-p` Prints disk partition table (`struct dkpart`).

`-q` Quiet mode – suppresses warning messages.

`-R` Configures `mroot` virtual disks (`mirror,root` and `mirror,swap`) in accordance with `/etc/dktab`. This configuration will ultimately be stored in the files `/mrconfig.1` and `/mrconfig.2`.

`-s` Console messages such as "vdisk237: Restoring previous state" are no longer output while mirror disks are being configured.

`-t` Prints disk type information (`struct dktype`).

`-u` Unconfigures a virtual disk.

`-v` Verbose mode.

`-w` Configures mirror disks. Catchup should not be started (only together with the `-c` option).

`-x` Prints this list.

`-z` Option to reset I/O statistics to zero for all pieces of all virtual disks.

The `-c` and `-u` options perform virtual disk configuration and unconfiguration. If the `-w` option is specified during configuration (only in the case of mirror disks) no Catchup is started for this mirror disk. If a Catchup is required, selected components (pieces) of the virtual disk must be activated or reactivated.

The `-l` option outputs the current configuration of the virtual disks in accordance with the contents of the configuration files.

The I/O statistics listed by the `-lv` option are used to determine whether I/O requests are distributed unevenly across physical disks.

The `-k` option prints configured virtual disks whether or not they are listed in the `/etc/dktab` file.

If disk pathname arguments are present, the requested operations (configure, unconfigure, list) are performed only for those disks specified. The `-a` option causes the specified option to be executed for all VL vdisk

groups created with the –L 0 option [see vdadmin(8)] as well as for all non-grouped virtual disks defined in /etc/dktab. If *all* the virtual disks defined in /etc/dktab are to be edited, this is possible using the –A option. If the –g option is used, the specified operation is executed for all disks defined in the group with the specified number. The –a, –A and –g options must not be combined.

dkconfig is normally silent when it is successful. If positive feedback is desired, the –v option should be specified. When listing the current virtual disk configuration, the –v option results in a more verbose listing where each segment of the virtual disk is described.

dkconfig does extensive error and cross-checking before configuring a virtual disk. Some of the warning messages can be suppressed with the –q option. The use of this option is not recommended as a general rule.

To allow for downward compatibility, the configuration files are also stored under /stand. If /stand is not a separate directory, however, they will be required under /.

EXAMPLES

The examples refer to operations using vdadmin(8) or /etc/dktab.

Configure all VL vdisks and all virtual disks defined in /etc/dktab:

```
/sbin/dkconfig –Ac
```

Configure all the virtual disks defined in /etc/dktab group 20:

```
/sbin/dkconfig –c –g20
```

Output overview of the current configuration of the VL vdisk groups created with the –L 0 option and the non-grouped virtual disks:

```
/sbin/dkconfig –al
```

Print a detailed list of the current virtual disk (grouped and non-grouped) configuration:

```
/sbin/dkconfig –Alv
```

Print a detailed description of virtual disk "vdisk2":

```
/sbin/dkconfig –lv /dev/vd/vdisk2
```

Print a short description of the first 4 virtual disks:

```
/sbin/dkconfig –l /dev/vd/vdisk[0–3]
```

Unconfigure all VL vdisks and all virtual disks defined in /etc/dktab:

```
/sbin/dkconfig –Au
```

DIAGNOSTICS

Most diagnostics printed by dkconfig are accompanied by the line number of the offending line in the /etc/dktab file. Line numbers are not output in the case of VL vdisks. In most cases, dkconfig continues with the next virtual disk in the configuration files after it has discovered a more or less fatal error.

`Cluster size (`N`) exceeds Length (`M`)`
: The cluster size of a striped virtual disk cannot exceed the length of any the virtual disk segments. The cluster size may be as small as 1 block and as large as the size of any segment. In addition, all segments of a striped virtual disk must be the same size.

`Unequal Lengths (`$N \neq M$`)`
: All segments of this virtual disk must be the same size.

`Length (`N`) exceeds physical partition size (`M`)`
: The size of a virtual disk segment is larger than the actual size of the physical disk partition.

`Length (`N`) is not a multiple of 32 blocks`
: Since most UFS file systems are a multiple of 32 blocks in length, dkconfig recommends, but does not require, that partitions be a multiple of 32 blocks.

`Offset (`N`) not on a cylinder boundary`
: The specified offset of a segment of a virtual disk does not appear to be on a cylinder boundary. This is only a warning that may be ignored if you think it is not important.

`Virtual disk type changed`
`Number of pieces changed`
`Cluster size changed`
`Catchup block size changed`
`Physical disk dev changed`
: The definition of an active virtual disk was changed. The printed description of the virtual disk is likely to be incorrect. The virtual disk in question should be closed and unconfigured before its definition in `/etc/dktab` is changed.

`Multiple declaration for ...`
: A particular virtual disk may only be defined once. You have defined a virtual disk more than once in `/etc/dktab` or in both `/etc/dktab` and the VL configuration files at the same time.

`Expected` *N* `piece definition lines`
`Missing virtual disk declaration line`
`Expected only` *N* `piece definition lines`
: The number of segments specified on the virtual disk declaration line does not match the number of physical disk lines that follow.

`Invalid number of pieces (maximum is` *N*`)`
: The virtual disk has too many segments.

`Rest of file ignored`
: An ignore keyword was encountered in the `/etc/dktab` file.

The remainder of diagnostics printed by `dkconfig` are self-explanatory.

NOTES

When using virtual disks, `/etc/dktab` represents a freely editable source of information in relation to the configuration of the virtual disks. It is therefore important to ensure the integrity of the `/etc/dktab` file at all times. The `vdadmin(8)` command is used to ensure the integrity of the VL configuration files.

FILES

`/dev/vd/vdisk[0–9]*`
: Block special virtual disks

`/dev/vd/rvdisk[0–9]*`
: Character special virtual disks

`/etc/daps/daps/vdisk.daps`
`/etc/daps/daps/vdiskgrp.daps`
: VL configuration file for virtual disks

`/etc/dktab`
: Virtual disk configuration file

SEE ALSO

`df(1)`, `fsck(1M)`, `mount(1M)`, `umount(1M)`, `dktab(4)`, `vfstab(4)`, `mroot(7)`, `vdisk(7)`, `vl(7)`, `vdadmin(8)`.

System Administrator's Reference Manual

dkindex(8)

NAME

`dkindex` - display/set disk index structures

SYNOPSIS

Format 1

`dkindex` `-l` *special*

Format 2

`dkindex` [`-w`] `-a` *c*`:`*h*`:`*s* `-c` *cyl special*

DESCRIPTION

Format 1 displays the data structure location index for disk *special*. The data structure location index is part of the PSDF (Pyramid Standard Disk Format) specification.

Format 2 sets the data structure location index for disk *special*. The following options are possible:

`-a` *c*`:`*h*`:`*s*

Specifies the location of the Format Information Table (FIT).

c Specifies the cylinder.

h Specifies the head.

s Specifies the sector.

`-c` *cyl*

Specifies the segment for disk-specific data such as the partition table.

`-w` Do not write index to disk (for drives with removable media).

special

The raw disk device to read from or write to.

Format 2 is generally only called from `dksetup`.

Each disk driver has a convention for locating the index structure on a particular disk. The index structure specifies where all of the other configuration data structures can be found.

EXAMPLES

```
# /sbin/dkindex -l /dev/ios0/rsdisk000s0
id: /dev/ios0/rsdisk000s10
asm             type            part            tune
1644:12:0       1645:0:4        1645:0:8        1645:0:12
-1:0:0          1645:1:4        1645:1:8        1645:1:12
-1:0:0          1645:2:4        1645:2:8        1645:2:12
-1:0:0          1645:3:4        1645:3:8        1645:3:12
-1:0:0          1645:4:4        1645:4:8        1645:4:12
-1:0:0          1645:5:4        1645:5:8        1645:5:12
-1:0:0          1645:6:4        1645:6:8        1645:6:12
-1:0:0          1645:7:4        1645:7:8        1645:7:12
```

SEE ALSO

`psdf`(4), `dkpart`(8), `dkpsdfinit`(8), `dksetup`(8), `dktune`(8), `format`(8) [RM600 only], `setinfo`(8).

System Administrator's Reference Manual

dkmap(8)

NAME

dkmap - print disk space usage map

SYNOPSIS

dkmap [−F *vfstab*] [−V *dktab*] [−bdiq] [−t afsruv] [*physical-block-device ...*]

dkmap [−F *vfstab*] [−V *dktab*] [−bdiq] [−a]

dkmap [−F *vfstab*] [−V *dktab*] [−bdiq] [−a *physical-block-device*]

dkmap [−F *vfstab*] [−V *dktab*] [−bdiq] [−a *physical-block-device length*]

dkmap [−F *vfstab*] [−V *dktab*] [−bdiq] [−a *physical-block-device length offset*]

DESCRIPTION

The dkmap command generates a disk space usage table for all connected disk drives [see autoconf(8)]. The type of usage is taken from the /etc/vfstab file, from the system database for Vdisk Lite [VL, see vl(7)] as well as the /etc/dktab file. This tool provides administrators with a complete summary of disk space use, reporting collectively on all physical disk space and how that space is used, whether by virtual disks, as file systems, swap space, or for an application-defined use. dkmap also checks for overlap of used disk space and prints warning indicators upon finding these errors.

dkmap is useful in determining the location of unused disk space, in addition to being helpful as a prevention tool; dkmap can be run prior to the mount(1M) , newfs(1M) , mkfs(1M) , or swap(1M) commands to check that disk space already in use is not going to be overwritten by the proposed changes.

If a partition or virtual disk is not used for file systems, but via the raw device interface, an entry must be made in vfstab. The character string raw is entered in the fstype field. If the entry is not made, the partition or virtual disk is displayed as being available (Available). You can also enter the character string avail in the fstype field of a vfstab entry, which implies that the partition is available.

Also note that any string within the fstype and the mountpoint fields of a vfstab entry need not be an existing file system or one of the above special strings. A system administrator may wish to use these fields to describe the use of that disk space in more specific terms than the generic string raw. For instance, if a partition is used by a data base for employee records, then the system administrator may wish to set fstype and mountpoint to be *database_name* and *employee*, respectively.

OPTIONS

−F Instead of using the system vfstab file (/etc/vfstab), use the specified file.

−V Instead of using the system virtual disk specification file (/etc/dktab), use the specified file. *Warning*: The system database for Vdisk Lite cannot be replaced.

−d Always show the disk offsets of the block range of each partition. Normally, the block range of the partition use is shown only when there appears to be a problem with the usage, or when the partition is available.

−b Always show the block ranges, but show them as relative offsets to the specified partition. Note that if there are overlapped usages of disk space, then this option may be ignored (this is because the disk offsets are shown for the offending block ranges).

−i Ignore size information from kernel and other sources. This option is particularly useful when reorganizing a disk, as it will suppress any error messages about inconsistencies in file system size between the old and new disk usage.

−q Quiet mode – suppress diagnostic messages.

−t Allows selective display of use types.

a Show available block ranges.

`f` Show block ranges that are used by file systems.

`r` Show block ranges that are used as a raw disk.

`s` Show block ranges that are used for swapping.

`u` Show block ranges that are inaccessible (see below).

`v` Show block ranges that are used as parts of a virtual disk.

`-a` This option is designed for use by a parsing program, hence the output is minimal and unformatted.

If zero, one or two arguments are given, the following output is produced:

`blkspecial=`*NNN*
`offset=`*NNN*
`length=`*NNN*

where `blkspecial` is the partition name containing the largest available disk block range for that disk device, `offset` is the offset of the beginning of the available blocks, and `length` is the number of available blocks in that range.

If no arguments are given, then the above description is given for each disk device with available space. If only the physical block device is given, then it is given for that device only. If the physical block device and the length are given, then it is given for the first available block range which will fit that many blocks. Note that this may be different from that shown by the other forms of this option.

If all three arguments, physical block device, length and offset are given, then `dkmap` will determine whether the use of that block range would cause any overlapped use of disk space. `dkmap` will not show the above format, rather it will use the output format described below. If no output is desired (just the exit code is required), then the `-q` option should be used.

If the `-a` option is not present and there are disk path name arguments present, then the display is limited only to those disks specified. Note, however, that `dkmap` still scans all disk partitions and all virtual disks encountered and determines the usage for each. This may produce warning messages for disks other than those specified.

DIAGNOSTICS

If the `-a` option is specified, then the exit value will be either `0` (zero) or `1` (one), based on 1) whether the specified use can be added to the system without any overlap or inconsistency or 2) whether a range of available space was found, depending on the number of arguments. If the `-a` option is not specified, then `dkmap` will exit with a value which is the bit-wise OR of the following:

`1` = Some disk space is available (unused).

`2` = An overlap in disk space usage.

`4` = There was at least one partition that had a questionable use, or at least one virtual disk that had a questionable configuration.

`8` = There was some fatal error.

If the `-a` option is specified, then the exit value corresponds only to the disks specified. Otherwise, it corresponds to all disks known to the system.

OUTPUT

The output of dkmap is organized in groups by physical disk. Each disk is output with a description of its use. Virtual disks follow with their usage. Each physical disk description has the following format:

disk_dkname (*size_of_disk_in* `Mb`):
flags *partition_path_name* *block_range* *usage*
...

Each virtual disk description has the following format:

flags *vdisk_dkname* *block_range usage*

flags have the following meaning and always appear in this order:

`?` the partition size or use is uncertain

`!` the partition use is overlapping with another use

`*` the partition is marked in `/etc/vfstab` as available but `dkmap` found a use for it

`@` the partition size is zero

`\~` the block range represents only a portion of the partition

block_range
starting and ending block offset of block range, can be either a disk offset, or be relative to the partition (it may not always appear)

usage has one of the following formats:

`AVAILABLE (`*number_of_blocks* `blks)`
`INACCESSIBLE (`*number_of_blocks* `blks)`
FSType`:`*mount_point*
vdisk_name`: piece` *vdisk_piece_number*
`<`*usage*`>` e.g. `<raw>` or `<swap>`

`INACCESSIBLE` block ranges are those that cannot be accessed via a disk partition. This is generally caused by either specifying an incorrect user-defined partition layout to `dkpart(8)` , or by a disk configuration error. Although the blocks are inaccessible to the system, they do not pose a problem and the error may be ignored (resulting in reduced disk capacity).

NOTES

`dkmap` is designed to detect any overlap or misconfiguration of partitions based on `/etc/vfstab`. It is a preventive, advisory measure to aid the administrator. A better solution would be a kernel-based implementation where partition usage would be rigorously enforced and overlapping uses prohibited.

The `/etc/vfstab` file represents the official source of disk usage information. `/etc/vfstab` file must be complete at all times. This is especially important if disk space is accessed only by the raw device interface.

`dkmap` can only be used when a virtual disk entered in `/etc/vfstab` is also described in the system database for Vdisk Lite or in `/etc/dktab`.

When using virtual disks, the VL configuration files as well as `/etc/dktab` represent the official source for information about the configuration of virtual disks. The integrity of the `/etc/dktab`file must therefore be guaranteed at all times.

MESSAGES

`NOTE: partition` *partition_name*`, used as` *use_description* `has 0 length.`
The specified partition has no length, hence this partition cannot be used and will be ignored.

`NOTE:` *vdisk_name* `is not configured in the kernel.`
`-- Using configuration from /etc/dktab`
The virtual disk has no kernel configuration. The related physical partitions will be marked with the flag `?`.

`NOTE: configuration for` *vdisk_name* `differs in kernel and in /etc/dktab`
`-- Using configuration from /etc/dktab`
The currently configured virtual disk does not match what is described in `/etc/dktab`. The related physical partitions will be marked with the flag `?`.

`NOTE: Don't know about FSType` *name*`.`
`Assuming FS uses entire length of` *partition_name*
`dkmap` does not know how to ascertain the size of the specified file system and thus assumes that it will use the entire disk partition. The related physical partitions will be marked with the flag `?`.

`NOTE:` *name* `can't find size of FS.`
`Assuming FS uses entire length of` *partition_name*
`dkmap` tried to ascertain the size of the specified file system but failed. It thus assumes that the file system will use the entire disk partition. The related physical partitions will be marked with the flag `?`.

`NOTE: currently swapping into partition` *partition_name*
: The kernel swap table indicates that this partition is being used, but there was no entry in `/etc/vfstab`.

`NOTE: swap partition '`*partition name*`' not known to kernel`
: The partition is marked for swapping, but the `swap(1M)` command has not been used to add that partition to the kernel swap table.

`NOTE: use of partition exceeds length`
`-- `*partition_name*`: len = `*N*`, use = `*N*
: The size of the used area of a partition is larger than the physical partition length. The related physical partitions will be marked with the flag `?`.

`NOTE: partition` *partition_name* `is marked as available`
`-- in /etc/vfstab but it is being used`
: The related physical partitions will have its usage field changed to `<?avail?>`.

`NOTE: can't find a partition for an unused block range`
`-- block range <`*N*`,`*M*`> on` *disk_name*
: While determining unused areas on the disk, `dkmap` could not find a partition that included the given block range.

FATAL ERROR MESSAGES

`Can't parse vdisk name (`*vdisk_name*`)`
: The path name of the virtual disk is not in the standard format (`/dev/vd/vdisk`*NN*).

`Cannot continue without virtual disk information`
: The library routine for parsing `/etc/dktab` returned an error. Check the consistency of `/etc/dktab` by running `dkconfig`.

`Cannot find physical disk (`*partition_name*`) for vdisk` *vdisk_name*
: Errors were encountered while trying to associate physical partitions with a virtual disk. Check the consistency of `/etc/dktab` by running `dkconfig`.

`Could not find a root file system entry`
partition_name `is not the root file system currently mounted`
: The root and swap partitions are determined automatically by the kernel, however, they should be defined in `/etc/vfstab`. `dkmap` (as with other file system and I/O utilities) ensures that these entries exist in `/etc/vfstab`. This is a particularly helpful diagnostic when booting from an alternative partition or when the kernel is configured to override the default root and swap partitions.

`Offset + length for exceeds partition size`
: The offset and length determined for the use of the partition exceeds the partition size.

`No disk found for partition being considered (`*partition_name*`)`
: The specified disk partition name does not correspond to any known disk on the system.

The remainder of the messages printed by `dkmap` are self-explanatory.

FILES

`/dev/CCCN/[r]Xdisk[0–9]*s[0–15]`
: physical disk device path names

`/dev/vd/[r]vdisk[0–9]*`
: virtual disk device path names

`/etc/daps/daps/vdisk.daps`
`/etc/daps/daps/vdiskgrp.daps`
: VL configuration files for virtual disks

`/etc/dktab`
: virtual disk configuration file

`/etc/vfstab`
: file system configuration file

SEE ALSO

swap(1M), dktab(4), vfstab(4), vdisk(7), vl(7), dkconfig(8), dkpart(8).

System Administrator's Reference Manual

dkmirror(8)

NAME

`dkmirror` - mirror disk administrative utility

SYNOPSIS

`/sbin/dkmirror` {–d*N*|–e*N*|–m*N*|–o*N*} [`–srlvwczN`] {*mirror_disk*|–g*M*}

`/sbin/dkmirror` `–C`*N mirror_disk*

`/sbin/dkmirror` {–d|–e|–m|–o} *autoconf-pathname*

`/sbin/dkmirror` `–D`{D|E} *autoconf-pathname*

`/sbin/dkmirror` `–a`[`v`] [–g*M*|–A|–O]

`/sbin/dkmirror` `–l`[`vc`] [–g*M*|–A|–O|*mirror_disk ...*]

`/sbin/dkmirror` `–L`

`/sbin/dkmirror` `–p`*N*

`/sbin/dkmirror` `–P`

`/sbin/dkmirror` {–E|–B|–R} *mirror_disk* [OPS only]

`/sbin/dkmirror` `–T`[`S`] [*io_size*] *mirror_disk*

`/sbin/dkmirror` `–S` [*speed_ratio*] *mirror_disk*

`/sbin/dkmirror` `–x`

DESCRIPTION

`dkmirror` is an administrative utility that manages the state of mirrored virtual and "statesave" disks that have already been configured via `dkconfig(8)` . Its functions include providing control over the state of individual mirror pieces, and reporting on the current state of mirror disks and their pieces. In addition, `dkmirror` can be used to suspend or resume I/O on a given mirror disk.

OPTIONS

`–A` Displays all mirror disks in all groups.

`–a` Sets all pieces of all specified (default: all non-grouped) mirror disks to "enabled", i.e. start all possible catch-ups.

`–B` Begins recovery mode.

`–C`*N* Switches piece *N* of the mirror disk directly from a `disabled` status (no catchup) to an `online` status. Observe the warning issued when using this option.

`–c` Displays all read and write accesses to the pieces.

`–D`{D|E}
Interactively disables (`D`) or enables (`E`) mirrors on a device.

`–d`*N* Disables mirror piece *N*.

`–E` Ends recovery mode.

`–e`*N* Enables mirror piece *N*.

`–g`*M* Lists or changes the current status of all mirror and "statesave" disks in group *M*.

`–L` Lists the current state of a mirror disk(s).

`–l` Same as `–L`, but alternative form of listing.

`–m`*N* Clarifies a piece of the *N* mirror disk as the main component (masterpiece).

`–N` (For `–d`*N* only): If the piece to be deactivated is the last available (online) piece of the relevant mirror

disk, it is not deactivated.

`-O` Displays all OPS mirror disks (with `-l`).

`-o`*N* Places mirror piece *N* online.

`-P` Catch-up priority query.

`-p`*N* Sets the priority of the catch-ups (*N*=1 ⇒ slow, but low system load, *N*=2 ⇒ average, *N*=3 ⇒ fast, but high system load). The catch-up priority cannot be changed if

- acatch-up is currently running. In such cases, the new priority is only set once all catch-ups are completed.
- OPS mirror disks are configured.

`-R` Restores the last piece disabled.

`-r` Resumes I/O on a mirror disk.

`-S` Displays or changes speed ratio [see section **MIRRORING USING FIBER OPTICS (FO)** towards the end of the text].

`-s` Suspends I/O on a mirror disk.

`-T` Defines speed ratio [see section **MIRRORING USING FIBER OPTICS (FO)** towards the end of the text].

`-v` Verbose mode.

`-w` Waits until catch-up is finished before exiting.

`-x` Displays syntax of `dkmirror`.

`-z` Resets the `-c` counter (see above) to 0. This option can be used only together with the `-c` option.

The `-d`, `-e`, `-m`, and `-o` options change the current state of one or more pieces of *mirror_disk*. Valid piece number arguments for these options are in the range of 0 to "the number of configured pieces -1". When the `-v` option is supplied with one of these options, positive confirmation is given for each piece state that was set.

Segment *N* of the mirror disk is clarified as a main component (masterpiece) using the `-m`*N* option. The next time the mirror disk is configured, a catch-up from this piece to the other will be started automatically.

Note: If `-g`*M* is specified rather than a virtual disk when `-demo`*N* is used, the operation is executed for all *N* pieces of the mirror disks in the group *M*.

Note: If no figure is specified for the `-d`, `-e`, `-m`, or `-o` option, but rather an "autoconf-pathname" of the form `iosx/sdiskxxx`, the required operation is carried out for all pieces of all mirror disks on the specified physical disk. If a SCSI port of the form `iosx/sportxx` is specified, the above is valid for all physical disks on this SCSI port.

The `-s` and `-r` options suspend and resume I/O on *mirror_disk*. When I/O is suspended on *mirror_disk*, all processes attempting to read from or write to it will block until a resume is done on *mirror_disk*. When obtaining a frozen copy of mirror disk data, these options aid the synchronization of pieces of a mirror disk prior to disabling one piece. Neither the root nor `/usr` partitions should be suspended.

The `-l` option lists the current state of a mirror disk(s). If *mirror_disk* pathname arguments are present, the states of those particular mirror disks are reported; otherwise, information is output about all mirror disks defined in the system database for Vdisk Lite as well as the mirror disks defined in `/etc/dktab`. When used with the `-v` option, piece state information is also displayed. See the **STATE OUTPUT** section for an explanation of these states. These options may be combined with any other options. When used with the `-g` option, only mirror disks and the "statesave" disk in the specified group are listed.

When the `-w` option is enabled, `dkmirror` does not exit until catch-up is finished. This can be useful in shell scripts for detecting catch-up completion. This option can be used with any of the other options, provided the command line contains exactly one *mirror_disk* pathname argument.

`dkmirror` establishes information about the configuration of the virtual disk by default from the system

database for Vdisk Lite as well as the `/etc/dktab` file. If the `DKTAB` environment variable is set, however, its value will be used instead of `/etc/dktab`. This capability of specifying an alternative `dktab` file exists for testing purposes only, and should not be used under any other circumstances.

`dkmirror` requires any *mirror_disk* pathname argument(s) to be a virtual block device.

For "statesave" disks only the options `–c`, `–d`, `–e`, `–l`, and `–v` are admissible.

EXAMPLES

Change the state of mirror disk `vdisk1` such that piece 0 is enabled and piece 1 is placed online:

```
/sbin/dkmirror –e0 –o1 /dev/vd/vdisk1
```

A catch-up process from piece 1 to piece 0 is started. □

Print the current state of all mirror default-grouped disks defined in `/etc/dktab`, including their pieces:

```
/sbin/dkmirror –lv □
```

Print the current state of all mirror disks and the "statesave" disk defined in `/etc/dktab` in group `20`, and their pieces:

```
/sbin/dkmirror –lv –g20 □
```

Suspend I/O on mirror disk `vdisk1`, disable piece 1, resume I/O on this `vdisk`, and then print its current mirror disk and piece state information:

```
/sbin/dkmirror –d1 –srlv /dev/vd/vdisk1
```

The following example is based on the following conditions:

- Partitions 1, 2, 3 and 4 of the `ios0/sdisk022` are used.
- There is an additional disk `ios0/sdisk042` of the same type with the same partition table.
- There is a requirement to mirror both disks together.
- The corresponding entries (four mirror disks) are recorded in the `/etc/dktab` file.
- The mirror disks were configured with `dkconfig`.

To activate the mirroring process, enter:

```
/sbin/dkmirror –o ios0/sdisk022
/sbin/dkmirror –e ios0/sdisk042
```

The catch-ups for the four mirror disks are started.

STATE OUTPUT FOR MIRROR DISKS

This section briefly describes several mirror disk and mirror piece states that are listed with the `–l` option. A further explanation of these states may be obtained from `mirror(7)` . Here is some sample output:

```
# dkmirror –lv
Statesave disk /dev/vd/vdisk10:     ONLINE (in use)
    piece  0  /dev/ios0/sdisk032s15    online
    piece  1  /dev/ios0/sdisk052s15    online
Mirror disk /dev/vd/vdisk11:        DOWN
    piece  0  /dev/ios0/sdisk032s1     disabled (default)
    piece  1  /dev/ios0/sdisk052s1     disabled (default)
Mirror disk /dev/vd/vdisk12:        NOT-MIRRORED (catch-up: 40%)
    piece  0  /dev/ios0/sdisk032s2     online   (catch-up source)
    piece  1  /dev/ios0/sdisk052s2     enabled  (catch-up in progress)
Mirror disk /dev/vd/vdisk13:        MIRRORED
    piece  0  /dev/ios0/sdisk032s3     online
    piece  1  /dev/ios0/sdisk052s3     online
Mirror disk /dev/vd/vdisk14:        DOWN
    piece  0  /dev/ios0/sdisk032s4     Master
    piece  1  /dev/ios0/sdisk052s4     enabled  (catch-up required)
# dkmirror –L
VDISK     TYPE STAT  |      PIECE 0          |       PIECE 1
---------------------+-----------------------+----------------------
```

```
                    | *** ios0/sport03 ***  | *** ios0/sport05 ***
vdisk10   Stat  ONL | ios0/sdisk032s15  onl | ios0/sdisk052s15  onl
vdisk11   Mirr  DOW | ios0/sdisk032s1   dis | ios0/sdisk052s1   dis
vdisk12   Mirr  NOT | ios0/sdisk032s2   onl | ios0/sdisk052s2   40%
vdisk13   Mirr  MIR | ios0/sdisk032s3   onl | ios0/sdisk052s3   onl
vdisk14   Mirr  DOW | ios0/sdisk032s4   Mas | ios0/sdisk052s4   ena
```

.fi

A mirror disk itself will be listed with one of these states:

`MIRRORED`
: Two or more mirror pieces are online.

`NOT-MIRRORED`
: Only one mirror piece is online.

`DOWN` No mirror pieces are online, rendering this mirror disk unusable.

In addition, a mirror disk state may be further described by one or both of the following:

`(catch-up: N%)`
: Catch-up is currently running, and is N percent complete.

`(suspended)`
: I/O has been suspended using the `-s` option.

If piece state information was requested with the `-v` option, each piece will be listed with one of these states:

`online`
: This piece contains valid data and can be used for reading and writing by the user.

`disabled`
: This piece is considered unusable; no user I/O will be attempted on this piece.

`enabled`
: This piece is eligible for use, and will become online once the validity of its data has been assured (either automatically by a catch-up or manually by a `dkmirror -o`).

`Master`
: The next time the mirror disk is configured, this piece is set to `online`. This status is reached using the `-m` option (primarily required for `mroot`).

Additional piece state information may also be given as detailed below.

If a piece is `online`, it may be further described by:

`(catch-up source)`
: Catch-up is running, and this piece is currently being read from as a catch-up source.

`(catch-up read error)`
: A catch-up read error occurred on this online piece; the piece remains online, but is not usable for catch-up until a redundant `-o` state change is made.

If a piece is `enabled`, the state of catch-up on this piece will also be listed:

`(catch-up in progress)`
: This piece is currently being caught-up.

`(catch-up pending)`
: This piece was enabled after a catch-up was already started; once that catch-up has finished, another catch-up will be started automatically to synchronize this piece.

`(catch-up required)`
: No online piece currently exists that could be used as a catch-up source for this piece.

If a piece is `disabled`, the reason it is currently disabled will also be supplied:

`(default)`
: `dkconfig(8)` has been run to configure this mirror disk, but no state has been set for this piece by `dkmirror`.

`(lost state)`
: The system was stopped abruptly (for example, a power failure occurred), and the state of this piece is now inconsistent.

`(I/O error)`
: There was an I/O error on this piece.

`(catch-up failed)`
: An attempt to bring this piece online by enabling it and allowing it to be caught-up automatically failed. Either there was a read error on the catch-up source or a write error on this piece.

`(incomplete catch-up)`
: A catch-up to this piece was interrupted by a system shutdown.

`(open failed)`
: An attempt to enable this piece failed due to an error from `open(2)` .

`(manual)`
: This piece was manually disabled via `dkmirror –d`.

STATE OUTPUT FOR "STATESAVE" DISKS

This section briefly describes several of the "statesave" disk and respective "statesave" disk piece states that can be listed with the `–l` option. Here is some sample output:

```
# dkmirror –lv
Statesave disk /dev/vd/vdisk1:            ONLINE (in use)
    piece  0  /dev/ios0/vdisk010s1        online
    piece  1  /dev/ios0/vdisk011s1        open
    piece  2  /dev/ios0/vdisk013s1        disabled (I/O error)
# dkmirror –lv –g3
Statesave disk /dev/vd/vdisk3:            DOWN
    piece  0  /dev/ios0/vdisk010s3        enabled
    piece  1  /dev/ios0/vdisk011s3        disabled (I/O error)
```

Note: In the above examples, `/dev/vd/vdisk1` was defined with "statsv", and `/dev/vd/vdisk3` was defined with "statesave".

A "statesave" disk itself will be listed with one of these states:

`ONLINE`
: Two or more "statesave" pieces are online.

`ENABLED`
: Only one "statesave" piece is enabled (meaning the read is successful and the write is yet to take place).

`DOWN` No "statesave" pieces are online or enabled, rendering this "statesave" disk unusable.

In addition, `(in use)` indicates that the disk is actively being used to store the state of configured mirror disk(s) and an attempt to unconfigure the "statesave" disk will fail.

If piece state information is requested with the `–v` option, each piece is listed with one of these states:

`online`
: This piece contains valid data and can be used for reading and writing the mirror disk states.

`disabled`
: This piece is considered unusable; no "statesave" I/O will be attempted on this piece.

`enabled`
: This piece is eligible for use, and the "statesave" read is successful. This piece will also be written during the next "statesave" update and will become `online` if the write is successful.

`open` This piece has been opened successfully; read/write is yet to take place. This piece will be written during the next "statesave" update and will become `online` if the write is successful.

Additional piece state information may also be given as detailed below.

If a piece is `disabled`, the reason it is currently disabled is also supplied:

`(I/O error)`
: There was an I/O error on this piece.

`(open failed)`
: An attempt to enable this piece failed due to an error from `open(2)` .

`(manual)`
: This piece was manually disabled via `dkmirror -d`.

NOTES

The definitions of active mirror virtual disks in `/etc/dktab` should not be altered while those mirror virtual disks are still configured. Changing active mirror disk entries in the `/etc/dktab` file may prevent one from obtaining a correct listing of current mirror disk states. This is because `dkmirror` derives virtual disk configuration information from the `/etc/dktab` file.

DIAGNOSTICS

"*mirror_disk* `has only` *N* configured pieces"
: When *mirror_disk* was configured with `dkconfig(8)` , only *N* pieces were defined in the `/etc/dktab` file. However, a piece number argument to one of the `-e`, `-d`, or `-o` options has been used that was greater than or equal to *N*.

"`Because another piece is already online, cannot set piece` *N* `online`"
: Only one piece can be placed manually online using the `-o` option. Any other pieces must be brought online by simply enabling them. This will cause an automatic catch-up to start which, once finished, will ensure these pieces contain identical data. At that time, the caught-up pieces will be placed online automatically by the catch-up daemon.

"`Piece` *N* `is already enabled`"
"`Piece` *N* `is already disabled`"
"`Piece` *N* `is already online`"
"`Piece` *N* `is already forced in error on next I/O`"
: Piece *N* is already in the state requested by one of the `-e`, `-d`, or `-o` options. These messages are for advisory purposes only; no adverse reaction is caused by requesting this redundant action.

"`An I/O error has occurred on piece` *N* `of` *mirror_disk*"
: Piece *N* was requested to be set to either the enable or the online state, but failed because of an I/O error.

"`Number of pieces in /etc/dktab does not match number currently configured`"
: The definition for the mirror disk being reported on has changed in the `/etc/dktab` file. In particular, fewer pieces are now defined.

The remainder of diagnostics printed by `dkmirror` are intended to be self-explanatory.

MIRRORING USING FIBER OPTIONS (FO)

If a mirror disk is in the state `MIRRORED`, a read job is processed by the fastest available piece. Write jobs must be sent anyway to all available pieces. Availability is estimated based on the length of the job queues of the physical disks being used. This estimate assumes that the physical media can process read jobs quickly and without delay.

This does not apply in the case of a hardware configuration, where a mirror piece is connected at a considerable distance using FO (up to 500 m). If, for example, twelve read jobs are issued for a mirror disk at the same time in this type of configuration, the jobs are distributed equally to both pieces. As a result, the fast piece finishes processing first, while the slow piece still has more jobs to process.

The `-S` and `-T` options have been introduced in order to take account of this difference in speed.

The `-T` option can be used to measure the speed ratio of the mirror pieces. The `dkmirror -T /dev/vd/vdisk37` command, for example, issues 100 read jobs with a size of 16KB to every "vdisk37" mirror piece and calculates the average duration of the jobs per piece. The output takes the form 100:82. This means as follows: In the time taken by piece□0 to process 100 jobs, piece□1 manages 82. This is what is

meant by the term speed ratio. The –T option allows a job size in the range 1K to 64K to be specified; the standard size is 16K.

Assume we have a /dev/vd/vdisk37 mirror disk, where piece□0 achieves double the data throughput, this ratio can be set with the dkmirror –S 100:50 /dev/vd/vdisk37 command. As a result, a read job to be processed with a weighting of 2:1 is assigned to piece□0. If twelve read jobs are now started for this "vdisk37", as in the above example, 8 jobs will be handled by piece□0 and 4 by piece□1. The overall throughput for the twelve jobs is thus improved.

The setting that is currently valid can be checked with dkmirror –S /dev/vd/vdisk37, and is output in the form 100:50. If the speed ratio was not set, 100:100 is output which means that both pieces are handled equally (as earlier).

The –T and –S options can be combined. Thus, for example, dkmirror –TS 4k /dev/vd/vdisk37 calculates the speed ratio for the 4K job size and enters it in the kernel structures.

Notes and restrictions:

1. Setting the speed ratio is part of the procedure of fine tuning a mirror configuration. There are no heuristics yet available in this regard.
2. dkmirror –T can only be executed if
   a) aminimum of five minutes has passed since the system was booted;
   b) all pieces of the mirror disk have the online status.
3. All settings made with dkmirror –S are lost when the mirror disk is unconfigured or when the system is rebooted. If necessary, the required jobs can be performed with dkmirror –S x:y by means of a start script. The system administrator is responsible for creating the start script. It should be noted that the start script must not contain any dkmirror –T ... commands.
4. Input for dkmirror –S x:y is set automatically to 100 and displayed. This means that the dkmirror –S 1:2 /dev/vd/vdisk37 command returns 50:100.
5. With the dkmirror –S 100:0 /dev/vd/vdisk37 command, piece□1 is no longer used for read jobs for "vdisk37".

FILES

/etc/daps/daps/vdisk.daps
/etc/daps/daps/vdiskgrp.daps
: VL configuration files for virtual disks

/etc/dktab
: virtual disk configuration file

SEE ALSO

dktab(4), mirror(7), mroot(7), vdisk(7), vl(7), autoconf(8), dkconfig(8).

System Administrator's Reference Manual

dkpart(8)

NAME

`dkpart -` maintain disk partition tables

SYNOPSIS

`dkpart –l` [`–b`] [`–n` *name*] {*device_name*|*text_file*}

`dkpart –e` [`–n` *name*] {[`–w`] *device_name*|*text_file*}

`dkpart` {`–d` {`std`|`root`|`lin`} |`–f` *file*} [`–n` *name*] [`–w`] *device_name*

`dkpart –g` *geo* [`–d` {`std`|`root`|`lin`}] [`–n` *name*] *text_file*

`dkpart –m` *plist device_name*

`dkpart –help`

DESCRIPTION

`dkpart` is used to maintain disk partition tables or to create disk partition tables for new disk types. To edit the partition table for an existing disk, `dkpart` is given the disk's character special *device_name*. Otherwise, `dkpart` may be used to create or edit a partition table stored in a *text_file* for a disk drive type.

Disk partitions are stored in a sixteen-element array of starting cylinder number and length. `dkpart` introduces a rule-based disk partitioning scheme which ensures changes to a given partition are reflected in other partitions. Thus, any change to any partition is properly reflected in the other partitions. This eliminates inadvertent overlaying of partitions. `dkpart` displays a map of the partitions on disk to further disclose any potential errors caused by overlaying partitions.

OPTIONS

`–b` Displays block size.

`–d` {`std`|`root`|`lin`}
: Sets general default partitions for *device_name* or writes these default partitions for *geo* to *text_file*. These default partitions are allocated as follows:

  `–d std`
  : Only partitions 7 and 15 are used, all other partitions remain unused (length 0).

  `–d root`
  : Sets standard partitioning for root disks.

  `–d lin`
  : Divides the entire disk into 15 partitions of equal size, where partition 15 always occupies the last cylinder.

`–e` Edits the partition table for a disk or a partition table stored in a text file.

`–f` *file*
: Sets partitions from *file* for *device_name*. *file* is the pathname of a text file. The format of this file must match the format created by `dkpart`. If *file* contains a "/", `dkpart` tries to open *file*, otherwise `dkpart` searches for *file* in the directory `/etc/default/diskinfo/dkpart`.

`–g` *geo*
: Calculates and edits partitions for given disk geometry. *geo* may be a character special device name of the "0" partition of a disk, a partition file or a disk geometry in the form

  `ncyls-reserved_cyls:nheads:nsectors:sectsize`

  where `ncyls` is the total number of cylinders, `reserved_cyls` is the number of cylinders to be reserved for alternate tracks and configuration information, `nheads` is the number of heads (tracks) per cylinder, `nsectors` is the number of sectors per head (track), and `sectsize` is the size in bytes of a sector.

`-l` Lists the partition table for a disk or a partition table stored in a text file.

In the case of disks, only the kernel data is output. The data is read from the disk with the first `open` or is set using `dkpart` *options*. In a "multi-hosted" environment, please note the clarifications for the `-X` option of the `dksetup` command [see dksetup(8)].

`-m` *plist*
The size of any partition *not* specified in *plist* is set to 0. *plist* is a comma-separated list of partitions, e.g. `-m10,0,1,2,3,4,5,15`. The partitions specified here must cover the disk completely without gaps or overlaps.

`-n` *name*
Sets the name of the partition table to *name*.

`-w` Does not attempt to write the partition table onto the media. This is the default for read-only media such as CD-ROM.

device_name
Character or block mode device name of disk partition "0" or the `autoconf` name.

text_file
The pathname of a text file.

`dkpart` is used to interactively edit the rules that define where a partition starts and ends. As rules are changed, `dkpart` shows the effect on partition sizes, layout and overlap. Once the user is satisfied with the changes to the partitions, the disk partition tables are updated.

Each partition has two rules: the first defines the starting cylinder, the second defines the ending cylinder. These rules may be:

`#` An absolute cylinder number (beginning with 0).

`#mb` A size in Mbytes. All Mbyte calculation in `dkpart` are in terms of 2^{20} or 1,048,576 bytes. All Mbyte calculations are rounded up, if necessary, to the next cylinder.

`>p`*X* An expression meaning the first cylinder after partition *X*.

`<p`*X* An expression meaning the beginning of partition *X*.

`=p`*X* An expression meaning the size of partition *X*.

`=p`*X*`/`*N*
An expression meaning 1/*N*th the size of partition *X*.

`=p`*X*`*`*N*
An expression meaning *N* times the size of partition *X*.

`$` An expression meaning the remainder of the disk.

`$/`*N* An expression meaning 1/*N*th the remainder of the disk.

`$/`*N*`*`*N*
An expression meaning the remainder of the disk rounded down to an integral multiple of *N* cylinders.

A partition size of 0 indicates the partition is not used.

The next examples show the standard rules (option `-d` {`std`|`root`|`lin`}) for splitting an HD04 into partitions:

Option `-d std`:

```
HD04 "default"  Geometry 17356-3:16:64 (cyls:heads:sectors)
Sector size = 512 bytes         8678.0 Total Mbytes (1MB = 1024 * 1024 bytes)
Partition  Start Rule   End Rule     First Cyl   Last Cyl  Cyl Count  Size (MB)
     0          0           0                   * * *    Unused    * * *
     1          0           0                   * * *    Unused    * * *
     2          0           0                   * * *    Unused    * * *
     3          0           0                   * * *    Unused    * * *
     4          0           0                   * * *    Unused    * * *
     5          0           0                   * * *    Unused    * * *
     6          0           0                   * * *    Unused    * * *
```

```
     7          0          <p15            0      17351      17352     8676.0
     8          0          0                   * * *    Unused    * * *
     9          0          0                   * * *    Unused    * * *
    10          0          0                   * * *    Unused    * * *
    11          0          0                   * * *    Unused    * * *
    12          0          0                   * * *    Unused    * * *
    13          0          0                   * * *    Unused    * * *
    14          0          0                   * * *    Unused    * * *
    15          17352      1           17352      17352          1        0.5
# Partition layout (not to scale):
|-----------------------------------7-----------------------------------|15|
```

.IP Option `-d root`:

```
HD04 "default" Geometry 17356-3:16:64 (cyls:heads:sectors)
Sector size = 512 bytes         8678.0 Total Mbytes (1MB = 1024 * 1024 bytes)
Partition  Start Rule   End Rule     First Cyl   Last Cyl  Cyl Count  Size (MB)
     0          >p10        256mb           8        519        512      256.0
     1          >p0         1024mb        520       2567       2048     1024.0
     2          >p1         512mb        2568       3591       1024      512.0
     3          >p2         512mb        3592       4615       1024      512.0
     4          >p3         512mb        4616       5639       1024      512.0
     5          >p4         <p6          5640       9159       3520     1760.0
     6          9160        <p15         9160      17351       8192     4096.0
     7          0           0                   * * *    Unused    * * *
     8          0           0                   * * *    Unused    * * *
     9          0           0                   * * *    Unused    * * *
    10          0           4mb              0          7          8        4.0
    11          0           0                   * * *    Unused    * * *
    12          0           0                   * * *    Unused    * * *
    13          0           0                   * * *    Unused    * * *
    14          0           0                   * * *    Unused    * * *
    15          17352       1           17352      17352          1        0.5
# Partition layout (not to scale):
|10|-0-|---1----|--2--|--3--|--4--|-----5------|------------6-------------|15|
```

.IP Option `-d lin`:

```
HD04 "default"  Geometry 17356-3:16:64 (cyls:heads:sectors)
Sector size = 512 bytes         8678.0 Total Mbytes (1MB = 1024 * 1024 bytes)
Partition  Start Rule   End Rule     First Cyl   Last Cyl  Cyl Count  Size (MB)
     0          0           $/15            0       1155       1156      578.0
     1          >p0         =p0          1156       2311       1156      578.0
     2          >p1         =p0          2312       3467       1156      578.0
     3          >p2         =p0          3468       4623       1156      578.0
     4          >p3         =p0          4624       5779       1156      578.0
     5          >p4         =p0          5780       6935       1156      578.0
     6          >p5         =p0          6936       8091       1156      578.0
     7          >p6         =p0          8092       9247       1156      578.0
     8          >p7         =p0          9248      10403       1156      578.0
     9          >p8         =p0         10404      11559       1156      578.0
    10          >p9         =p0         11560      12715       1156      578.0
    11          >p10        =p0         12716      13871       1156      578.0
    12          >p11        =p0         13872      15027       1156      578.0
    13          >p12        =p0         15028      16183       1156      578.0
    14          >p13        <p15        16184      17351       1168      584.0
    15          17352       1           17352      17352          1        0.5
# Partition layout (not to scale):
|-0--|-1--|-2--|-3--|-4--|-5--|-6--|-7--|-8--|-9--|10--|11--|12--|13--|14--|15|
```

The first lines always contain the name of the partition table (HD04), the disk geometry and the size of the disk in Mbytes. The next line contains the headings of the partition division rules and partition sizes. For each

partition, a rule is specified for the beginning of the partition, end of the partition, start cylinder, end cylinder, as well as its size in cylinders and Mbytes. The partition overview is followed by a disk allocation diagram which illustrates how the partitions are arranged and any overlapping partitions. The disk allocation diagram shows the size of the partitions in relative terms and not to scale.

In the second example (root partitioning of HD04), partition 10 starts with cylinder 0 and is approx. 4 MBytes long (rounded to a full cylinder). This partition is provided for `boot2`. The start of partition 0 can be specified at cylinder 8 or `>p10` (i.e. following partition 10). Thus, partitions 0, 1, 2, 3, 4 and 5 follow partitions 10, 0, 1, 2, 3 and 4 (`>p10`, `>p0`, `>p1`, `>p2`, `>p3`, `>p3`) and have a length of 512, 2048, 1024, 1024, 1024 and 3520 cylinders or approx. 256, 1024, 512, 512 and 1760 Mbytes. Partition 6 starts after partition 5 (`>p5`) and ends before partition 15 (`<p15`) and has a length of 4096 MByte. Partition 15 has a length of one cylinder and is intended to be a `statesave` device for mirroring disks. Partitions 0, 1, 2, 3, 4 and 5 are provided for `/`, `/swap`, `/opt`, `/usr`, `/var` and `/home`.

When editing the partitions of a disk, only the start and end rules need to be changed. As the start or end rule is changed, the offsets and lengths of all other partitions and the partition layout map are recalculated.

The editor uses the following commands:

`h`, `j`, `k`, `l`
: Move left, down, up or right one rule. The terminal arrow keys (if any) may be substituted.

`n` Change the name of this partition table.

`TAB` Move right one field.

`s` Save changes.

`w` Write partition to text file.

`q` Quit.

`u` Undo changes to field.

`U` Undo all changes.

`D` Calculate standard partition size for root disk partitioning.

`?` Display help about `dkpart` commands.

`r` Display help about partition rules.

EXAMPLES

Display the partitions of disk `ios0/rsdisk000`:

```
dkpart –l ios0/rsdisk000s0
```

Edit the partition table of disk `ios0/rsdisk001`:

```
dkpart –e ios0/rsdisk001s0
```

Edit the partition table of the file `part_file`:

```
dkpart –e part_file
```

You use `dkpart` as follows to configure standard partitioning for a root disk with 17356 cylinders, 3 reserved cylinders, 16 tracks per cylinder, 64 sectors per track and 512-byte sectors:

```
dkpart –d root –g 17356-3:16:64:512 -n NEWDISK text_file
```

NOTES

It is not possible for `dkpart` to detect the existence of useful information contained on a disk, nor will it back up or recover any existing data. It is the user's responsibility to back up data, remake file systems and/or reload databases.

`dkpart` is normally called by `dksetup(8)` to write disk partitioning data into the PSDF (Pyramid Standard Disk Format). `dkpart` generates default partitioning based on the disk geometry.

FILES

`/etc/default/diskinfo/dkpart/*`

SEE ALSO

psdf(4), dksetup(8), dktune(8), format(8), setinfo(8).

System Administrator's Reference Manual

dkpsdfinit(8)

NAME

`dkpsdfinit` - initialize disk data structures

SYNOPSIS

`dkpsdfinit` [–v] *device_path*

DESCRIPTION

`dkpsdfinit` initializes the PSDF (Pyramid Standard Disk Format) data structures on the disk named *device_path*. *device_path* should be raw disk partition zero. For example, `/dev/ios0/rsdisk000s0` specifies partition zero of the first IOS (SCSI) disk on the system.

The disk type, partition information, and tuning parameters currently in use are written to the disk. Since there is no change in the operating parameters of the disk, this command is always safe to run.

The data structures will be written into a reserved area of the disk that must have previously been specified using `dkindex`, `dksetup`, or `format`.

Use `setinfo`, `dkpart`, and `dktune` to modify the operating parameters of the disk.

OPTIONS

–v Verbose. The progress of the command is noted on `stdout`.

DIAGNOSTICS

Exit status is normally `0`. If an error occurs, exit status is non-zero.

SEE ALSO

`psdf`(4), `dkindex`(8), `dkpart`(8), `dksetup`(8), `dktune`(8), `format`(8) [RM600 only], `setinfo`(8).

System Administrator's Reference Manual

dksetup(8)

NAME

`dksetup` - initialize/update physical disk configuration

SYNOPSIS

`dksetup –U` *devname* [*disk_type*]

`dksetup –I` *devname* [*disk_type*]

`dksetup –F` *devname* [*disk_type*]

`dksetup –N` *devname*

`dksetup –R` *devname*

`dksetup –X` *devname*

`dksetup –L` *devname* "*string*"

DESCRIPTION

`dksetup` initializes a new disk or updates an existing disk for use with the current version of the system software. `dksetup` is a simplified high level interface to the low level disk administration commands. Disk administration performed by `dksetup` includes:

1. building nodes in `/dev`
2. formatting disk media
3. initializing disk configuration data structures (`dkindex`, `dkpart`, `dktune`, `setinfo`)
4. writing the "Format Information Table" (FIT), if not available (`format`)
5. standardizing the internal disk parameters in the same way as for the write/read cache and for error handling, if this was not already done at an earlier stage (`setinfo`)

devname is the generic name of a disk device as displayed by `autoconf`(8) . For example, `ios0/sdisk000` is the first IOS (SCSI) disk on the system. Device names of the type `/dev/ios0/rsdisk012s0` may be used as well.

disk_type is a specification of the disk manufacturer's model number. The known type specifications are the names of files in the `/etc/default/diskinfo/setinfo` directory.

If *disk_type* is not specified, it is assumed that it is a disk not recognized by the system, i.e. there is no corresponding file in the `/etc/default/diskinfo/setinfo` directory. In this case, the disk is initialized with standard parameters.

Specific administration tasks such as modifying disk partitioning, forwarding bad blocks, performance tuning, or building file systems may be done using the individual low-level commands `dkpart`, `format`, `dktune`, and `mkfs`.

OPTIONS

`–U` Upgrades a disk for the current release of the system software. This is non-destructive. It is always safe to run, even if the disk is being used. This initialization only takes place when it is determined that it has not yet been, or was only partially carried out. The significance of this option has been restricted in order to prevent certain attributes, such as the status of the write/read cache or the current operating mode, from being changed while the system is running.

`–I` Initialization as with the `–U` option is always carried out, but is only required for the *first* initialization of a disk.

`–F` Formats/initializes a new disk. Initializing a disk may include formatting of the media and writing default configuration parameters onto the disk. This is always **DESTRUCTIVE**. Great caution should be exercised to ensure that any useful data on the disk has been saved.

`-N` Builds the nodes of a new disk in `/dev`. This option should be used for removable disks, like ROD and WORM drives. For removable disks, default disk configuration data structures are calculated on first open. This option may be used even if no medium is in the drive.

`-R` Upgrades the LUN disk device specified. The other actions of option `-U` will be performed.

Note:

This option does only work with the optional `sraid` driver, and will fail with the standard `sdisk` driver.

`-X` Is only intended for a multi-hosted environment. If *host1* changes the hard disk configuration (e.g. the partitioning), then *host1* can use the command "`rsh dksetup -X` *devname*" to tell *host2* to read the current configuration the next time this disk is accessed.

`-L` This option allows you to change the name of the partition. This name is displayed when `dkpart(8)` and `autoconf(8)` are called. It is thus possible, for example, to change the `autoconf` display of the root disk from

```
ios0/sdisk001 MP85 -- MP85
```

to

```
ios0/sdisk001 MP85 -- Root Disk Y543B00.
```

For some I/O subsystems, it may be necessary to run `dksetup -F` on a disk to properly initialize it for the system. This is particularly true when a disk is moved over from an OSx (TOS) system to Reliant UNIX. SCSI disks in particular require `dksetup -F` to be run to re-install the Format Information Table on the disk. The option to install the IOS Format Information Table [see `format(8)` for RM600] is non-destructive. Refer to `sdisk(7)` for more information.

FILES

```
/etc/default/diskinfo/setinfo
/etc/default/diskinfo/dktune
```

SEE ALSO

`mkfs(1M)`, `psdf(4)`, `autoconf(8)`, `dkindex(8)`, `dkpart(8)`, `dktune(8)`, `format(8)` [RM600 only], `setinfo(8)`.

System Administrator's Reference Manual

dktune(8)

NAME

`dktune` - read/write disk tuning information

SYNOPSIS

`dktune –l` *pathname*

`dktune –d [ –wv ]` *pathname*

`dktune –t [ –wv ]` *tunefile pathname*

`dktune [ –wv ]` *pathname* < *tunefile*

DESCRIPTION

The `dktune` utility is used to initialize, update or display the `dktune` structure in the PSDF [Pyramid Standard Disk Format – see `psdf(4)`]. `dktune` is normally called with a tuning structure supplied by Pyramid, prior to formatting a new disk.

After formatting, dktune may be used to change or display the tuning structure.

Tuning data structures are files stored in `/etc/default/diskinfo/dktune/*`.

OPTIONS

`–d` Sets the disk tuning to a generic default (for example, no disk tuning and little optimization).

`–l` Lists tuning parameters from the disk.

`–v` Lists tuning parameters after setting.

`–w` Does not attempt to write the tuning structure on disk.

`–t` *tunefile*
: Sets the disk tuning as specified in *tunefile*. If *tunefile* contains a "/", `dktune` tries to open *tunefile*, otherwise `dktune` opens *tunefile* in the directory `/etc/default/diskinfo/dktune`.

pathname
: Device path name.

EXAMPLES

Use the following to display the tuning data from the PSDF for disk `/dev/ios/sdisk000s0`:

```
dktune –l /dev/ios/rsdisk000s0
```

The output is in the `dktune` source format:

```
##dktune for /dev/ios0/rsdisk000s0
#
          dku_flags='DKU_ELEVATOR'
        dku_trkskew=13       # physical head skew
        dku_cylskew=21       # physical cylinder skew
         dku_rotgap=8        # rotational gap between all requests
         dku_trkgap=0        # rotational gap on track switches
         dku_cylgap=0        # linear disk head acceleration on seeks
       dku_rotdelay=2        # file system rotational delay in milliseconds
       dku_velocity=0        # max linear disk head velocity on seeks
  dku_timeout_high=500000    # IO_high priority queue timeout
  dku_timeout_norm=700000    # IO_normal priority queue timeout
   dku_timeout_low=1000000   # IO_low priority queue timeout
   dku_timeout_ord=0         # IO_ordered priority queue timeout
```

Use the following to initialize or update PSDF on `/dev/ios0/sdisk000` with the `dktune` data from `/etc/default/diskinfo/dktune/HP97536`:

```
dktune –t HP97536 /dev/ios0/rsdisk000s0
```

To initialize or update PSDF on `/dev/ios0/sdisk000` with the default tuning structure:

```
dktune –d /dev/ios0/rsdisk000s0
```

NOTE

It is better to specify default (no) tuning than to load incorrect tuning data.

FILES

`/etc/default/diskinfo/dktune/*`

SEE ALSO

`dkpart`(8), `dkpsdfinit`(8), `format`(8) [RM600 only], `setinfo`(8).

System Administrator's Reference Manual

eip2cmd(8)
RM400 only

NAME

`eip2cmd -` issue commands to the EIP2

SYNOPSIS

`/etc/opt/havail/eip2cmd [ –v ]` *cmd-string*

DESCRIPTION

`eip2cmd` allows the system administrator to issue commands to EIP2 (Environmental Interface Processor) at the application level. This requires that the caller is familiar with the EIP2 commands.

The EIP2 command (*cmd-string*) is entered without the "NL" character. This character is added by the command.

Numbers within the EIP2 command (*cmd-string*) are interpreted as hexadecimal numbers.

The EIP2 response is output as text to `stdout` and contains the prompt character "`>`" at the end of the text. The "EIP2-RESPONSE:" is added to this by the command.

The EIP2 command (*cmd-string*) is transferred with the added "NL" character to the system (`ioctl`) and is written unchanged by the system to the V.24 line to the EIP2. The response from the EIP2 is passed transparently up to the application level.

OPTIONS

`–v` Instruction for verbose output to `stdout`.

EXAMPLES

Input: `eip2cmd getrel 2`

Output: `EIP2-RESPONSE: '2000>'`

Input: `eip2cmd getstat 2`

Output: `EIP2-RESPONSE: '0042>'`

Input: `eip2cmd iw 2 20 5`

Output: `EIP2-RESPONSE: '50>'`

DIAGNOSTICS

The `eip2cmd` command returns the same response text as EIP2 (see document specified above). Incorrect commands are responded to with the text "!00>" or "?*cmd-string*".

NOTES

The EIP2 command `getint` is not permitted and is rejected. The interrupt status register of the EIP2 in the system cabinet is always read by the system, if an interrupt was issued by the EIP2 in the system cabinet.

The command ends with exit code 0. If there are errors in the `ioctl` call, the exit code is –1, the error output to `stderr` contains the specific `errno` value [see `ioctl(2)`].

SEE ALSO

`switchscu2(8)` .

System Administrator's Reference Manual

eipd(8)
RM400 only

NAME

`eipd` - monitor system cabinet

SYNOPSIS

`/usr/sbin/eipd`

DESCRIPTION

The `eipd` daemon monitors the system cabinet as well as all peripherals cabinets with EIP.

The `eipd` daemon is controlled by means of the `/var/adm/ubft/_upstab` file [see `_upstab`(4)]. It reacts, in particular, to the following events:

error ::= {`acf2` | `fan_unit_fail` | `temp_fail` | `acf1` | `eip_lost` | `too_few_ps_units`}

SEE ALSO

`_upstab`(4), `ptfd`(8), `pupsd`(8).

System Administrator's Reference Manual

errlogd(8)
RM600 only

NAME

`errlogd -` threshold value monitoring for correctable device errors

SYNOPSIS

`errlogd`

DESCRIPTION

The Errlog daemon has the following characteristics:

1) It reads the component limit files (CL files) and initializes the internal data structures.
2) It reads the error data from the error daemon.
3) It performs threshold value monitoring.
4) It generates the corresponding error messages for Logging V3.0.

Re 1) The Errlog daemon is started via the `/etc/init.d/errlogd` startup script when the system is powered up. An important condition for the startup of threshold value monitoring is the availability of a component limit file (CL) for each hardware component to be monitored. This file contains all of the limit values and sense data which are valid for a hardware component. The CL files must be present in the `/etc/default/diskinfo/errlogd` directory during runtime.

The format of the CL file is:

<*sense_data*> <*reference_unit*> <*soft_limit*> <*hard_limit*> <*additional_information_text*>

Where:

<*sense_data*> = <*sense_key*> <*additional_sense_code*> <*sense_code_qualifier*>

The elements "sense_key", "additional_sense_code" and "sense_code_qualifier" are each 2 bytes long.

Formats of the "reference_unit":

Time format:

<*name_of_time_unit*><*number_of_time_units*>

Data format:

`B:`<*number_of_transferred_bits*>

Order format:

`O:`<*number_of_orders*>

A distinction is made between the following time unit names "name_of_time_unit": "`D:`" for a day, "`H:`" for an hour, "`M:`" for a minute, and "`S:`" for a second. The elements of the time format must not be separated. The individual elements of the CL file must be separated by a blank or a tab. If the first character of the line in the CL file is a "#" or a blank, the entire line is regarded as a comment. The soft and hard limit are positive numbers. The concrete values for the sense data and for soft and hard limits can be found in the technical specifications provided by the vendor for the individual hardware components. The "additional_information_text" is freely selectable and can be up to 32 bytes long.

For the hardware components being monitored, the name of the CL file must consist of the readable characters of the vendor and product names in the *ad_inqname* element of the `struct autoconf` data structure.

Format of the CL file name:

<*readable_characters_of_vendor_name*>_<*readable_characters_of_product_name*>

Threshold value monitoring is activated for those hardware components for which there is a CL file available.

Re 2) After threshold value monitoring has started, the error data is read from the error daemon via a named pipe.

Re 3) The threshold value analysis is performed with regard to the number of errors that occur within a given time period (number of transferred bits or number of orders). The state of the hardware component can be derived from the ratio of errors to limits.

Error/time	<	Soft limit			=>	`normal` state
Soft limit	<=	Error/time	<	Hard limit	=>	`not critical` state
Hard limit	<=	Error/time			=>	`critical` state

Re 4) The `normal` state is not reported to Logging V3.0. The `not critical` state is reported to Logging V3.0 as a "soft error message" and the `critical` state as a "hard error message". In the case of the soft error message, the component attribute is set to `ERROR` in the `HW_CONFIG` file. In the case of the hard error message, the component attribute is set to `LAR_DEFECT` in the `HW_CONFIG` file. Once the hard error message has occurred, no further error messages are reported to Logging V3.0, and threshold value monitoring is switched off for this hardware component. Monitoring of the components can be recommenced after the error daemon is restarted, or using the "`set_errlogd -r`*hardware_component_name*" command. Only hard disks are monitored at present.

Format of the error messages for the hard disks:

"Soft error message":

`Error entity:`
`EE_DISK`

`Error number:`
`1`

Error severity:
`LOG_WARNING`

`System component:`
`ios0/sdisk`*xxy*

`Target:`
<*Controller_ID*><*SCSI_ID*>

`Soft limit value:`
First message: Value of the soft limit in plain text.
All subsequent messages: Number of the message (>1)

`Sense data:`
Extended sense key
Additional sense code
Additional sense code qualifier

`Sense data means:`
Statement about the sense data

"Hard error message":

`Error entity:`
`EE_DISK`

`Error number:`
`2`

`Error severity:`
`LOG_CRIT`

`System component:`
`ios0/sdisk`*xxy*

`Target:`
<*Controller_ID*><*SCSI_ID*>

`Hard limit value:`
Value of the hard limit in plain text

`Sense data:`
Extended sense key
Additional sense code
Additional sense code qualifier

`Add. Inform. Text:`
Text from the component limit file

Comment:

xx =Controller_ID: Scope `00 – ff`

y =SCSI_ID: Scope `0 – f`

In addition, the internal error messages (e.□g. the error messages of the system calls) are forwarded to Logging V3.0.

Format of the error messages from the Errlog daemon:

`Error entity:`
`EE_ERRLOGD`

`Error number:`
0

`Error severity:`
0

`Error text:`
Statement about the cause of the error

FILES

`/etc/init.d/errlogd`
"Startup" script

`/sbin/errlogd`
Errlog daemon

`/sbin/set_errlogd`
Command for controlling the Errlog daemon

`/etc/default/diskinfo/errlogd/`*CL-files*
Component limit files

`/opt/errlogd/errlog.cnf`
Configuration file

`/opt/errlogd/errlog.sav`
Backup file

`/opt/errlogd/erlg_pipe`
Pipe from `errord` to `errlogd`

`/opt/errlogd/user_pipe`
Pipe from `set_errlord` to `errlogd`

SEE ALSO

`errord`(8), `set_errlogd`(1M), `logger`(1).

System Administrator's Reference Manual

errord(8)

NAME

`errord` - log system state and errors

SYNOPSIS

`/sbin/errord`

DESCRIPTION

`errord` performs the following functions:

1. reads system log entries from `/dev/error`
2. formats, compresses, and saves the entries in `/var/adm/error/errfile.current.Z`

Since `errord` compresses its output file, it cannot append to an existing error file. If `errord` is started and `/var/adm/error/errfile.current.Z` already exists, it will be moved to `/var/adm/error/errfile.previous.Z`.

`errord` is normally invoked at boot time by the `/etc/init.d/errord` system startup script. This script saves old error files with names of the form `/var/adm/error/errfile.`*MM-DD-hh:mm*`.Z` where *MM*, *DD*, *hh*, and *mm* is the month, day, hour, and minute.

Error log entries are written to the log file in a fixed format designed to be both readable in itself and easily parsed for further processing. A formatted entry consists of two sections. The header contains pairs (identifiers, value), and the data section contains a simple hexadecimal dump of operating system structures that fully describe the error condition.

The header identifiers are:

`len` The number of bytes of data in the `DATA` section of the log entry.

`time` The time at which the entry was logged.

`type` A high level classification of the error.

`ET_NONE`
`ET_GOTS`
: system startup message

`ET_STOP`
: system stop message

`ET_SOFT`
: asoftware condition caused this entry

`ET_HARD`
: ahardware condition caused this entry

`ET_MEDIA`
: adisk or tape media fault caused this log entry. This type of entry contains a fixed format `DATA` section so that generic fault analysis software may examine the error histories.

`severity`
: An indication of just how bad things are.

`ES_NONE`
`ES_NOTE`
: informational

`ES_WARN`
: warning, better pay attention

`ES_ERROR`
: recoverable error

`ES_PANIC`

the subsystem described has to become unavailable

`ES_CONT`
continuation of a previous message

`ES_COR`
correctable media fault

`ES_UNCOR`
uncorrectable media fault

`entity`
A general description of what produced the log entry.

`EE_NONE`
`EE_DISK`
disk controller/media

`EE_TAPE`
tape controller/media

`EE_CTLR`
other controller/adaptor

`name` A specific device or subsystem name.

`prog` The name of and arguments to the log entry formatters that analyze and format the log entry.

`desc` A text string describing the log entry.

FILES

`/dev/error`
pseudo-device giving access to kernel error log

`/var/adm/error/errfile.current.Z`
error log file

`/var/adm/error/errfile.previous.Z`
previous error log file

`/etc/init.d/errord`
system startup script

`/var/adm/error/bin`
directory containing log entry formatters

SEE ALSO

`compress`(1), `zcat`(1), `error`(7), `pyr_iosall`(8).

System Administrator's Reference Manual

fdsetup(8)
RM600 only

NAME

`fdsetup` - initialize/update physical floppy configuration

SYNOPSIS

`fdsetup –U` *devname* [*type*]

DESCRIPTION

`fdsetup` initializes a new floppy or updates an existing floppy for use with the current version of the system software. `fdsetup` is a simplified high level interface to the low level floppy administration commands. Floppy administration performed by `fdsetup` consists of building the correct device nodes in `/dev`.

devname is the generic name of a floppy device as displayed by `autoconf`. For example, `ios0/sfdisk005` is the floppy drive SCSI-ID 5 on logical controller 0. Refer to `devname(7)` for more information.

`–U` upgrades a floppy drive for the current release of the system software. This is non-destructive to the floppy.

SEE ALSO

`devname(7)`, `autoconf(8)`.

System Administrator's Reference Manual

flup(8)

NAME

`flup` - load new online firmware for controller or SCSI devices (online Flash Utility Program)

SYNOPSIS

`/sbin/flup` [[–[`a`]`c`] | [–[`a`]`s` [–`f` *parameterfile*]]]

DESCRIPTION

`flup` is a menu controlled utility program that can supply SCSI devices with new firmware versions. In addition to this, a firmware update of the flashable controller is also possible.

OPTIONS

`–a` Automatic updating. All controllers or SCSI devices in the system with old firmware are defined and automatically loaded with the new version.

WARNING:
It is not possible to intervene or terminate the program manually.

`–c` Controller update (menu-driven).

`–s` SCSI device update (menu-driven).

`–f` Determines the parameter file for the SCSI device update. (The parameter selection menu is bypassed.)

Prerequisites:

The online flash utility supports the following HW and SW systems for the SCSI flup:

RM200/300/400
SINIX-N from version 5.42 B00; Reliant UNIX from version 5.44 A00.

RM600-xxx
SINIX-Y from version 5.42 A30; Reliant UNIX-Y from version 5.43 C00.

Where SCSI devices are to be flashed on operating systems that have earlier versions than those listed above, this is only possible for special releases.

Flashing of controllers is only possible on the RM600-xxx with operating system versions from SINIX-Y 5.41 A00 onwards.

WARNING:
Newly loaded firmware only becomes active *following the next reboot* of the machine (the SCSI bus simply needs to be reset when SCSI devices are updated).

INSTALLATION

Installation of CD-ROM:

1. Insert the Reliant UNIX operating system CD-ROM in the CD-ROM drive.
2. As superuser `root` enter the `cdinst` command.
3. Choose the CD-ROM drive.
4. Choose the `SIfirm` package from the product list.
5. Choose the `install` menu option.

Following installation, change to the `/home/flup` directory.

START

Before you start `flup`, the firmware files, and also the parameter files in the case of SCSI flup, must be available in the `/home/flup` directory. The files for the controller update are installed automatically when the

`SIfirm` package is installed. Individual corrections and all data for the SCSI update are provided as required by means of the SCR (single correctional release) server and must be copied to `/home/flup`.

If `flup` is invoked without parameters, you are first shown a menu where you can choose initially between SCSI and controller flash.

`flup` is fully menu-driven. The most important function keys are the cursor up and cursor down keys and the first 8 function keys. A Help function can be invoked for every menu with <F1> (alternatively <h> or <?>), which provides information on how to use the respective menu.

The execution is divided into two phases:

A) The *selection phase*

B) The *action phase*

Controller flup (manual operation)

Call:

```
# flup
```

or:

```
# flup –c
```

A) The *selection phase* of the controller flash

1. Controller selection

   Once the system configuration has been checked, which may take a varying amount of time depending on how the machine is configured, a table is output of all controllers in the system that are in principle flup-enabled.

   If there are more controllers in the system than screen lines, it is possible to scroll through the display using the <PageUp> and <PageDown> keys or the relevant function keys <F2> and <F3>.

   A controller that cannot be updated is identified by a "#" preceding the relevant line; the reason for this is given at the end of the line. Such a controller cannot be selected for update.

   To select an entry, use the arrow keys to move the cursor to the required entry and press the selection key <F4> or <x>. The selected entry is now highlighted as such by the "*" character at the start of the line. You can select any number of controllers in this way. Inadvertently selected entries can be canceled by selecting them again.

   To exit this selection menu, press <F5> or <q>.

   WARNING:

   DCS controllers on the RM600-xxx are only displayed in single-user mode.

   You can select <F6> or <a> to choose all flup-enabled controllers for update at the same time. This selection menu is then exited immediately; no further input (e.g. removing individual controllers) is then possible.

   The <F7> or <u> key automatically causes all flup-enabled controllers to be loaded immediately with the latest firmeware without any further input. This is the same reaction as when calling `flup –ac`.

   When this menu has been exited with <F5> or <F6> following selection of the respective controllers, you enter the next stage of selection, i.e. the firmware selection.

2. Selection of the controller firmware

   All controllers chosen for update are listed in a selection window specifying the controller name, the cabinet, slot, bus ID and hardware version.

   The firmware currently active on the controller and that selected for the flash is displayed in two additional columns. Initially no firmware is selected for the flash, this is displayed using "??.??".

   Now move the selection cursor successively to each controller entry and select it with <F4> or <x>. A list of firmware versions available for this board is then shown in another window. The minimum hardware version required for the respective firmware is also shown in a second column. If the hardware version of the controller is lower than the minimum version required, the relevant firmware is

marked as not available for selection by a "#" at the beginning of the line.

Now select one of the available firmware versions by pressing the <F4> or <x> key. This firmware version is now shown in the firmware selection window instead of "??.??". You can exit this menu without making a selection by pressing <F5>. The selection window can be invoked any number of times.

The <F7> key can be used to assign the latest suitable firmware to each controller at the same time

Once you have processed all the selected boards in this way, confirm your selection using <F5> or <q>. The action phase starts.

WARNING:

Before ending the selection phase, make sure a firmware version is entered in the "newFW" field for all controllers. If this entry is not present, the controller is not included in the action phase.

B) The *action phase* of the controller flash

In the action phase, the firmware versions chosen in the selection phase are transferred to the controllers. You can now no longer intervene in the program execution.

Progress reports are provided on each screen for each controller for the following program phases:

- Split phase
  Splitting of the firmware files into parts fitting the individual EEPROMs. Display of the firmware checksums.
- Deletion phase
  Deletion of the EEPROMS from the controller.
- Programming phase
  Loading of the EEPROMS of the controller with the new firmware.

`flup` is ended automatically once the action phase is completed.

SCSI device flup (manual operation)

Call:

```
# flup
```

or:

```
# flup –s
```

or:

```
# flup –s –f parameterfile
```

A) The *selection phase* of the SCSI device flash

1. Selection of the parameter files
   If you call `flup` without specifying a parameter file, firstly a selection window opens which displays an overview of all the parameter files available in the system, from which you can select just one for each flash procedure.

   Each parameter file contains all the important information specific to the update of a SCSI device. The name is selected in such a way that reference to the device and to the firmware to be loaded is recognizable (e.g. `MP81_FW9466.par` for firmware version 9466 of the MP81 hard disk).

   Select the required file using <F3> or <x> and confirm your selection using <F5> or <q>.
2. The SCSI device selection menu
   All the SCSI devices corresponding to the selected parameter file are listed in a selection window (e.g. all MP81 hard disks). You can now select as many devices as you require for flashing, as described in point 1. Then you terminate the selection phase using <F5> or <q>.

   If all devices are selected, press <F4> or <a>. All devices will be selected and the flash procedure will be started immediately without further input confirmation.

B) The *action phase* of the SCSI device flash

All the SCSI devices selected in the selection phase are loaded *in sequence* with the firmware defined in the parameter file. Manual intervention by the user is no longer possible at this point.

During the flash procedure, `flup` provides the following progress reports in lines 18 to 21:

Line 18:
: The foreseeable total runtime of all device updates.

Line 20:
: The device currently being processed.

Line 21:
: Detailed reports on the individual action phases.

Once the flash procedure is completed for a device, a success report is output in line 21 and the next device is processed.

Once all the devices are exhausted, the action phase and `flup` are terminated automatically.

System Administrator's Reference Manual

format(8)
RM600 only

NAME

`format` - online disk formatter on an RM600

SYNOPSIS

`/sbin/format`

DESCRIPTION

`/sbin/format` is a shell script used to format disk drives. It requires no arguments and it prompts for all the parameters it needs.

Supported IOS SCSI disks are self-configuring, so no system configuration information [via `setinfo(8)`] is necessary. Unsupported devices may be configured using `setinfo(8)` . Defective sectors are mapped and saved by the SCSI disk controller. The `Pyramid Standard Disk Format` (PSDF) data structures are written onto the disk at the completion of the format. A Format Information Table is also written on the disk, which contains the time of last format, and drive serial number information. Any time the Format Information Table is printed, the SCSI defect list is read and printed as well.

Since the SCSI format command involves no system I/O, the command may be run at any time. The disk will be unavailable for use until the format completes.

The format process may take well over an hour depending on system load and the capacity of the disk. The program concludes with the format information table. The alternate format information table will be referred to as the defect table in this document.

The operations available are:

Initial Complete Format

Completely removes all data from the disk starting from the last cylinder finishing at the first cylinder. Initializes defect table, and maps any defects found in the new defect table. This should be used when a newly bought disk is added to the system. (Option `a`)

Reformat

Completely removes all data from the disk, sparing the defect table. Adds any defects found to the defect table. This should be used to erase a disk and check for new defects. (Option `b`)

SCSI Format

Executes the SCSI format command on the IOS disk. This completely removes all data from the disk and initializes the defect table. This should be used to initially format media or to reformat SCSI disks. Once the SCSI format command is started, it cannot be stopped! The SCSI format command time varies with the disk drive and can range from five to fifty minutes. (Option `f`)

Map Individual Tracks

The user is prompted for the starting physical block, and number of blocks to map. (Option `m`)

Install IOS Format Information Table

Initializes the IOS Format Information Table, and prints the table along with the SCSI defect list from the disk. This option allows preformatted IOS drives to be initialized without the need of reformatting the drive once again in order to install the information table. (Option `s`)

Read-only Scan

Does not remove any data from the disk. Reads every track without using ecc correction. Adds any defects to the defect table, and salvages data from the defect to the remapped track or sector. This can be used

periodically to detect defects from the gradual deterioration of the disk media. (Option `c`)

Partial Disk Format

This option can be used to format or read scan portion of the disk. User is prompted for the starting and ending physical block. (Option `p`)

Severe Burn-in

Completely removes all data from the disk, sparing the defect table. Uses several data patterns which detect disk defects differently. Adds any defect found to the defect table using a low error threshold. This can be used to insure all defects have been found. It will take several hours. (Options `d`, `e`)

Print Alternate Track or Format Information Table

Does not remove any data from the disk. Prints out information on the size of the drive, its serial number and lists the cylinder and head locations of every mapped track or sector. This can be used any time. (Option `T`)

The `t` option outputs the number of alternate tracks as opposed to the actual tracks.

NOTES

The `format` program does not check to see if the drive to be formatted is in use. The backend commands handle this and will exit if the drive if needed.

The `format` program makes no checks to see if the file given in `$FMTLOG` by a user exists or is writable.

FILES

`ios?/sdisk???`
: IOS disk device

`/var/adm/fmtlog`
: copy of console log during format

If the environmental variable `$FMTLOG` is set, then `format` will attempt to write the console log to the file indicated by this variable. Otherwise, `format` will look for a writable file among the following: `/var/adm/fmtlog`, `/usr/tmp/fmtlog`, `/tmp/fmtlog`, or `/dev/null`. This last possibility ensures that `format` will indeed find a writable file for the console log.

SEE ALSO

`dkindex`(8), `dkpsdfinit`(8), `setinfo`(8).

System Administrator's Reference Manual

getty(8)

NAME

`getty` - set terminal type, modes, speed, and line discipline

SYNOPSIS

`/sbin/getty` [`–h`] [`–t` *timeout*] *line* [*speed* [*type* [*linedisc*]]]

`/sbin/getty –c` *file*

DESCRIPTION

`getty` is a program that is invoked by `init(1M)`. It is the second process in the series, (`init-getty-login-shell`) that ultimately connects a user with the UNIX system. It can only be executed by the superuser, that is, by a process with the user ID of `root`. Initially `getty` prints the login message field for the entry it is using from `/etc/gettydefs`. `getty` reads the user's login name and invokes the `login(1)` command with the user's name as an argument. While reading the name, `getty` attempts to adapt the system to the speed and type of terminal being used. It does this by using the options and arguments specified in the command line.

The two files `/usr/lib/terminfo` and `/etc/termtab` are then read, to get the initialization strings and country code for `keyload(1M)` [97801].

OPTIONS

line The name of a TTY line in `/dev` to which `getty` is to attach itself. `getty` uses this string as the name of a file in the `/dev` directory to open for reading and writing.

`–h` If the `–h` flag is not set, a hangup will be forced by setting the speed to zero before setting the speed to the default or specified speed.

`–t` *timeout*
Specifies that `getty` should exit if the open on the line succeeds and no one types anything in *timeout* seconds.

speed
The *speed* argument is a label to a speed and TTY definition in the `/etc/gettydefs` file. This definition tells `getty` at what speed to run initially, what the login message should look like, what the initial TTY settings are, and what speed to try next, should the user indicate, by pressing the BREAK key, that the speed is inappropriate. The default *speed* is 300 baud.

type The *type* argument is a character string describing to `getty` what type of terminal is connected to the line in question. `getty` recognizes the following types:

`none` default

`ds40-1`
Dataspeed40/1

`tektronix`, `tek`
Tektronix

`vt61` DEC vt61

`vt100`
DEC vt100

`hp45` Hewlett-Packard 45

`c100` Concept 100

The default terminal is `none`; i.e. any CRT or normal terminal unknown to the system. Also, for terminal types to have any meaning, the virtual terminal handlers must be compiled into the operating system. They are available, but not compiled in the default state.

linedisc

The *linedisc* argument is a character string describing which line discipline to use when communicating with the terminal. Again, the hooks for line disciplines are available in the operating system but only one is presently available, i.e. the default line discipline `LDISC0`.

`-c` *file*

When `getty` is invoked with the `-c` option and *file*, it scans the file as if it were scanning `/etc/gettydefs` and sends the results to standard output. If there are any unrecognized modes or improperly constructed entries, it reports these. If the entries are correct, it prints out the values of the various flags. See `ioctl(2)` to interpret the values. Note that some values are added to the flags automatically.

When given no optional arguments, `getty` specifies the following: The *speed* of the interface is set to 300 baud, either parity is allowed, newline characters are converted to carriage return line feed, and tab expansion is performed on the standard output. `getty` types the login prompt before reading the user's name, one character at a time. If a null character (or framing error) is received, it is assumed to be the result of the user hitting the BREAK key. This will cause `getty` to attempt the next *speed* in the series. The series that `getty` tries is determined by what it finds in `/etc/gettydefs`.

After the user's name has been typed in, it is terminated by a newline or carriage return character. The latter results in the system being set to handle carriage returns appropriately [see `ioctl(2)`].

The user's name is scanned to see if it contains any lowercase alphabetic characters; if not, and if the name is non-empty, the system is told to map any future uppercase characters to the corresponding lowercase characters.

Finally, `login` is `exec`'d with the user's name as an argument. Additional arguments may be typed after the login name. These are passed to `login`, which will place them in the environment [see `login(1)`].

`getty` logs all errors via `syslogd(1M)` using `auth.crit`.

NOTES

While `getty` understands simple single character quoting conventions, it is not possible to quote certain special control characters used by `getty`. Thus, you cannot log in via `getty` and type a `#`, `@`, `/`, `!`, `_`, backspace, `^U`, `^D`, or `&` as part of your login name or arguments. `getty` uses them to determine when the end of the line has been reached, which protocol is being used, and what the erase character is. They will always be interpreted as having their special meaning.

FILES

`/etc/gettydefs`
`/etc/default/login`
`/etc/termtab`
`/etc/terminfo`

SEE ALSO

`ct(1)`, `login(1)`, `init(1M)`, `ioctl(2)`, `gettydefs(4)`, `inittab(4)`, `login(4)`, `termtab(4)`, `terminfo(4)`, `tty(7)`.

System Administrator's Reference Manual

ikdb(8)
RM600 only

NAME

ikdb - internal kernel debugging facility

SYNOPSIS

cmd [*parameters*]

DESCRIPTION

ikdb is an optional facility that can be built (i.e. linked) into the kernel and that runs in the kernel environment. During the initial stages of bringing up a system, the debugger should be configured into the kernel, so that it will be available to help diagnose system problems. Once the system is stable, you can deconfigure the debugger to save space in the kernel.

To configure or remove the debugger, edit the file /etc/conf/sdevice.d/ikdb:

```
ikdb Y 1 0 0 0 0 0 0
```

The second field (Y or N) is the configure parameter. If it is set to Y, the debugger will be linked into the kernel. If the field is set to N, the debugger is excluded from the kernel.

This configuration takes effect the next time you rebuild the kernel.

ikdb can be entered in several ways:

1. ikdb is called from several key points in kernel code:
   a) as early as possible in the startup process. This can be disabled/enabled by PCOS/SASH.
   b) during execution of an exec call immediately prior to entering user code. This can be enabled/disabled by the on/off ikdb commands.
   c) from panic to allow for postmortem analysis. This can be disabled/enabled by PCOS/SASH.
2. ikdb may be entered asynchronously by entering the DEBUG_INTERRUPT_STRING defined in ikdb/space.c on the system console. By default this string is set to "\0IKDB" (to get "\0" you have to type CTRL-@).
3. ikdb is entered as the result of encountering a breakpoint established by ikdb.

ikdb maintains the notion of a "current process" and a "current stack". On entry, the current process is that process on behalf of which the system is now executing (i.e. the "owner" of the current user structure). The current process can be changed to another process by the ne or cp commands so that processes that are currently dormant can be manipulated. The ne and cp commands also change the current stack to that of the new current process.

ikdb commands have the following form:

command parameters

where *command* is a command (described below) and *parameters* are parameters or options to be applied to the command. If several parameters are enclosed in square brackets ([□]), they are optional, and zero or more can be specified. If several parameters are enclosed in braces ({□}), exactly one must be specified. The output of most commands is stopped every 20 lines to allow the operator the option of terminating the command. The backspace key deletes the last character typed and CTRL-U willl delete the entire command line.

Numeric expressions

Numeric expressions use a syntax compatible with the C programming language and consist of arbitrary combinations of numeric strings, kernel variables and arithmetic operators (+ - * / < > & | ^ ~). The arithmetic operators have the same meaning as they do in C. An additional operator (@) is available to specify the value found at a particular virtual address. Placing a numeric expression inside square brackets

([*expression*]) has the same effect as the indirect operator * in C.

Expressions are evaluated left to right with operator precedence as in C. In addition, parentheses can be used to control the order of evaluation. The indirect operator can be used with additional indirect operators to achieve multiple levels of indirect accesses.

Numeric constants are evaluated as base 16 numbers unless preceded by the prefix 0b for binary (base 2), 0 or 0o for octal (base 8), 0t for decimal (base 10), or 0x for hex (base 16). The default base can also be changed with the cb command.

Symbolic addressing

Global kernel symbols can be used in any numeric expression to specify the address associated with the variable or procedure name. Kernel symbols are denoted by prepending the underscore character (_) to the symbol name.

COMMAND SYNTAX AND DESCRIPTION

Display/modify memory commands

da [{*address* | n | l} [*count*]]
dc [{*address* | n | l} [*count*]]
dh [{*address* | n | l} [*count*]]
dl [{*address* | n | l} [*count*]]

If *count* is negative, then memory is displayed backwards from *address*.

ds [{*address* | n | l} [*count*]]

Display memory contents in ascii (da), as disassembled code (dc), in hex (dh), as kernel symbols (dl), or as a string (ds). These are simply aliases for the sm command.

mm [{*address* | n | l} [*value*]]

The modify memory command is used to change the contents of the specified memory location. *address* is a numeric expression with the result being used as the address of the memory location to be modified. Specifying n or l for the address has the same meaning as for the display commands. If numeric expression *value* is given, the result is put into memory.

If no value is specified, an interactive memory editor is started. The editor will display the specified address and the contents of the memory location at that address. The value can be changed simply by entering a new value. The editor will then display the next address and contents of memory at that address. To end the process, enter a dot (".") for the value. Entering a dash ("-") for the value will decrement the address and continue the editing process. The memory editor accepts only constants; numeric expressions are not permitted.

sm [{-a | -i | -l | -s | -x}] [{*address* | n | l} [*count*]]

The show memory command displays the contents of memory in ascii (-a), as disassembled instructions (-i), as kernel symbols (-l), as a string (-s), or in hex (-x). Display starts at the address specified by *address* and continues for *count* words. If *address* is not specified, the address beyond the end of the last display command is used. The start address can also be specified with respect to the last display command (n or l). Using n indicates that the next address should be used allowing the display to continue on from where the previous one left off. To reuse the previous start address, specify the address as l.

The default count of 4 is used if the parameter *count* is missing. When formatting as a string, the display will terminate whenever the count has been reached or a NULL character is encountered. A non existent or zero count for strings will display memory until a NULL is encountered.

un Undo the last modify memory command. A modification has a global lifetime. Hence the previous mm command can have occurred in a previous entry to ikdb. Also note that this will only undo the last change of a sequence of modifications made in the memory editor.

Display/modify context commands

cp *cpu*

The processor executing `ikdb` is changed to the new *cpu*. This also has the side effect of changing the current (and original) process to the one currently active on the new *cpu*.

`mtf` {`s` | `r`}
By default, this command displays a stack trace for all cpus in the system. With the `s` option, a stack trace is displayed for all sleeping processes. With the `r` option, a stack trace is displayed for all running processes.

`ne` [[`p`] *process*]
The debugger environment is changed to reflect the state of the specified process. If no processes are given, `ikdb` reverts back to its original context. *process* can be either the address of the process table entry, or if the `p` option is specified, the process ID of the new process.

`np` *cpu*
Run the debugger on *cpu*.

`regs` [`-a`] [`-x`] [*address*] [*reg*[`=`*val*]]
The `regs` command displays the contents of the processor's registers. The `-a` option displays the contents before the last panic or assertion failure. The display width of the register contents is set to 64 bits by specifying the `-x` option. Supplying an optional *address* displays the registers in the exception frame at that location. The entire register set is displayed unless a specific register is designated by *reg*. If the numeric expression `=`*val* is specified, the contents of the specified register are changed to *val*.

`fregs` Display the floating point registers.

`sf` *level*
The `sf` command sets the registers and the state of `ikdb` to the specified stack frame, making it look as though execution had been interrupted at that point.

`tf` [*levels*] [*sp pc*]
This command displays a trace stack frames. For each frame on the stack, the function name, instruction address, first four parameters, and the stack frame address are printed. If *levels* is specified, the trace only proceeds for the number of stack frames specified rather than tracing back to the beginning of the stack. The trace starts from the current top of stack unless the *sp* and *pc* parameters are given.

Breakpoint and single step commands

`bp` [`-a`] *address* [*count*]
Set a breakpoint at the specified *address*. If *address* is at the beginning of a function, the breakpoint is placed at the end of the basic block instruction preamble generated by the compiler. This can be overridden using the `-a` option. Execution of the kernel stops and `ikdb` is restarted after the breakpoint is encountered *count* times. The *count* value defaults to 1.

`B` List all breakpoints currently set.

`d` *index*
Delete the breakpoint specified by *index*, where *index* is the index associated with the breakpoint as shown in the list of breakpoints (see the `B` command).

`D` Delete all breakpoints currently set.

`lb` List all breakpoints currently set.

`so` [*count*]
Similar to the `st` command except that function calls are executed as a single step.

`st` [*count*]
Execute *count* (defaults to 1) machine instructions.

`w` [`-rw`] [`-p` *pattern*] [`-m` *mask*] [*address*]
Without parameters this command shows the watchpoint currently set. With the `-rw` option a watchpoint can be specified at address for reading (`-r`), writing (`-w`) or both. Address must be from K0SEG to K1SEG.

The watchpoint is pattern sensitive for each bit which is set to 1 by the `-m` option. It is then compared to the appropriate bit in the pattern specified by the `-p` option.

`W` Delete watchpoint currently set.

Process display/control commands

`fp` {`p` | `c` | `w` | `s` | `t`} *regexp*

Find and list all processes that match a specific selection criterion. Exactly one flag must be given to indicate the field for the search: process ID (`p`), command (`c`), wait channel (`w`), process state (`s`) or terminal name (`t`). *regexp* is a regular expression used to compare against the given field. For example:

```
fp c ^get
```

Would find and list all processes executing commands with names beginning with "get".

`ki` [`-s`] [*-sig*] [*pid*]

Send a signal to a process. If specified, *sig* specifies the signal to send; 9 is the default. *pid* indicates the process to signal. If *pid* is missing, then all processes except system processes are signaled. System processes cannot be killed unless the `-s` option is specified and the process ID is explicitly stated.

`lp` The `lp` command lists all the processes active in the system and gives a short status description. The processes that are currently running on a CPU are highlighted.

`wa` *wchan*

Perform the Reliant UNIX wakeup call on the specified wait channel. *wchan* must be specified; there are no defaults.

Data structure display commands

This group of commands prints the contents of data structures. For most of these commands, the structure can be specified by the following arguments:

{*address* | `offset`}

If *address* is given, the command prints the name of each member of the structure (truncated to 8 characters), and the member's value, as in the following example for `percpu`:

```
(lcpu 0, pcpu 1) ikdb percpu c0030000
c0030000:
xclbind         0 idleproc c03aa400 lcpunum         1 lcpumask        2
percpufl        7 runrun           1 kprunrun        0 curpri         4f
npwakecn        0 dosoftin         1 dotimein        0 dqgen           1
migrate_        0 intrcxt        100 locktestoff     0 lastlock       ff
locklist [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0] 
idleflag        0 backoff          5 idleadj         0 baseprio        0
fp_proc         0 curproc  c0739c00 old_curp c03aa400 tlbpid_a       29
tlbpid_g       ca curipl           0 maxqueip        0 iplquema e0118000
iplfree  c00479ec iplbase  c0004000
```

If `offset` is specified, the printout specifies the full name of each structure member, its offset from the beginning of the structure in bytes, and its size in bytes. The following is an example of the `percpu` command:

```
(lcpu 0, pcpu 1) ikdb percpu offset
member(offset,size)
xclbind(0,4)        idleproc(4,4)       lcpunum(8,1)       lcpumask(c,4)
percpuflags(10,4)   runrun(20,4)        kprunrun(24,4)     curpri(28,4)
npwakecnt(2c,4)     dosoftint(30,4)     dotimein(34,4)     dqgen(38,4)
migrate_from(3c,4)  intrcxt(60,2)       locktestoff(62,1)  lastlock(63,1)
locklist(64,7c)     idleflag(e0,4)      backoff(e4,4)      idleadj(f0,4)
tlbpid_assigned(12c,4) tlbpid_gen(130,4) curipl(134,4)     maxqueipl(138,4)
iplquemask(13c,4)   iplfree(140,4)      iplbase(144,4)
```

`arp` Displays the contents of all `arptab` structures.

`as` {*address* | `offset`}
Displays the contents of a particular `as` structure.

`buf` {*address* | `offset`}
Displays the contents of a particular `buf` structure.

`cg` {*address* | `offset`}
Displays the contents of a particular `cg` structure.

`datab` {*address* | `offset`}
Displays the contents of a particular `datab` structure.

`dintr` {*address* | *number*}
Displays *number* of `dintr` structures beginning at *address*. Default value of *number* is `1`.

`fifonode` {*address* | `offset`}
Displays the contents of a particular `fifonode` structure.

`hat` {*address* | `offset`}
Displays the contents of a particular `hat` structure.

`hat_data` {*address* | `offset`}
Displays the contents of a particular `hat_data` structure.

`inode` {*address* | `offset`}
Displays the contents of a particular `inode` structure.

`intrtab` {*address* | *number*}
Displays *number* of `intrtab` structures beginning at *address*. Default value of *number* is `1`.

`iosconf` [*address*]
Displays the IOS configuration. If the *address* is specified, the IOS configuration of the specified port is displayed (address of a `shaimaster` structure).

`lpb` {*address* | `offset`}
Displays the contents of a particular `lpb` structure.

`lpbfree` *sha-address*
Displays the contents of `lpb` structures from the "freelist" administration, beginning with the address from "sha_lpbtail".

`lpbpdq` [`-a`] *pdq-address*
Displays the contents of `lpb` structures of a particular `pdq` structure. If the `-a` option is specified, only the active `lpb` structures are displayed.

`lpbsortq` [`-bs`] *pdq-address*
Displays the contents of `lpb` structures in a `sortq` structure. If the `-b` option is specified, the `buf` structures are output instead of the `lpb` structures. The `-s` option gives an overview, that is, only the addresses are output rather than the contents of the structures.

`lpbstat` *addr*
Displays the status of a particular `lpb`.

`lpbtim` *sha-address*
Displays all `lpb` structures of the shaimaster *sha-address* which have active timers.

`msgb` {*address* | `offset`}
Displays the contents of a particular `msgb` structure.

`np_adm` {*address* | `offset`}
Displays the contents of a particular `np_polladm` structure.

`np_dat` {*address* | `offset`}
Displays the contents of a particular `np_polldat` structure.

`page` {*address* | `offset`}
Displays the contents of a particular `page` structure.

`pdq` {*address* | `offset`}
: Displays the contents of a particular `shaipdq` structure.

`percpu` {{*lcpu* | *address*} | `offset`}
: Displays the contents of the `percpu` structure for a particular processor. The processor is specified by either its logical CPU number (*lcpu*) or the address of its `percpu` structure (*address*). If `offset` is given, then the offset of each member into the structure is printed as described above.

  If no arguments are given, this command prints the values for the current active CPU's `percpu` structure (note the active `lcpu` number in the prompt) with the structure member names truncated to 8 characters.

`pp` {{*pid* | *address*} | `offset`}
: Displays the contents (in logical order) of the `proc` structure for a particular process. The process is specified by either its process ID (*pid*) or the address of its process table entry (*address*).

`ppa` {{*pid* | *address*} | `offset`}
: Displays the contents (in alphabetical order) of the `proc` structure for a particular process. The process is specified by either its process ID (*pid*) or the address of its process table entry (*address*). If `offset` is specified, then the offset into the structure of each field is printed.

`qobj` *addr*
: Displays the contents of a particular `qu_runobj` structure.

`qpsync` *addr*
: Displays the contents of a particular `qp_sync` structure.

`queue` {*address* | `offset`}
: Displays the contents of a particular `queue` structure.

`rnode` {*address* | `offset`}
: Displays the contents of a particular `rnode` structure.

`rtable`
: Displays the contents of the NFS `rtable`.

`sdi` {*address* | `offset`}
: Displays the contents of a particular `sdiinfo` structure.

`seg` {*address* | `offset`}
: Displays the contents of a particular `seg` structure.

`sha` {*address* | `offset`}
: Displays the contents of a particular `shaimaster` structure.

`snode` {*address* | `offset`}
: Displays the contents of a particular `snode` structure.

`sortq` {*address* | `offset`}
: Displays the contents of a particular `dksortq` structure.

`sti` {*address* | `offset`}
: Displays the contents of a particular `stiinfo` structure.

`stream` {*address* | `offset`}
: Displays the contents of a particular `stream` structure.

`strtty` *address*
: Displays the contents of a particular `strtty` structure.

`tco` {*address* | `offset`}
: Displays the contents of a particular `tco_endpt` structure.

`tim_tim` {*address* | `offset`}
: Displays the contents of a particular `tim_tim` structure.

`user` {*address* | `offset`}
: Displays the contents of a particular `user` structure. The current `user` structure is displayed if no

parameter is specified.

`var` [`offset`]
Displays the contents of the `var` structure (v).

`vnode` {*address* | `offset`}
Displays the contents of a particular `vnode` structure.

`vfs` {*address* | `offset`}
Displays the contents of a particular `vfs` structure.

`vxau` *address*
Displays the contents of a particular `auheader` structure.

`vxaus` *address*
Displays the contents of a particular `vx_ausum` structure.

`vxctran` *address*
Displays the contents of a particular `vx_ctran` structure.

`vxea` *address*
Displays the contents of a particular `extalloc` structure.

`vxext` *address*
Displays the contents of a particular `extent` structure.

`vxfs` *address*
Displays the contents of a particular `fs` structure (vxfs).

`vxifree`
Displays the free list of the `vxfs` inodes.

`vxino` *address*
Displays the contents of a particular inode structure (vxfs).

`vxlog` *address*
Displays the contents of a particular `vx_log` structure.

`vxmap` *address*
Displays the contents of a particular `vx_map` structure.

`vxmem` *address*
Displays the contents of a particular `kmp` structure.

`vxmlink` *address*
Displays the contents of a particular `vx_mlink` structure.

`vxral` *address*
Displays the contents of a particular `vx_realloc` structure.

`vxrorg` *address*
Displays the contents of a particular `vx_extreorg` structure.

`vxtran` *address*
Displays the contents of a particular `vx_tran` structure.

`vxvfs` *address*
Displays the contents of a particular `vx_vfs` structure.

Virtual memory translation commands

`pt` *address*
Display the translation information associated with the given virtual address. This includes the physical address, and protection flags from both the kernel's data structures and the hardware TLB (Translation Lookaside Buffer).

`td` [`-c`] [{`-l` | `-s`}] [*index* ...]
Display the contents of the TLB. When `ikdb` is started, the contents of the TLB are saved and used for this command. To use the current contents of the TLB, specify the `-c` option. To dump specific TLB locations, specify the indices of the locations to be dumped. The data is output in the standard format

(`-s`, default) or in the old output format (`-l`).

Assertion Commands

`assert` [`-s` *sym*] *type* [*action* |*mode_bits* | *size*]]

Enables or disables assertions.

When you specify the `-s` option, the symbolic name of the assertion structure segment can be specified using *sym* (default: `_kern_assert`).

type is the assertion type (`all` stands for all types).

mode_bits is a mask that can be used to set the assertion mode. Valid modes are:

`1` Assertions with minimal loss of performance.

`2` Assertions that create status information and login information.

`3` Assertions that do not backup any data but influence performance.

or any possible combination of these modes (Example: 1 + 2 + 4 or 7).

The following can be specified for *action*:

`create`
Creates a ring buffer

`free` Frees a ring buffer

`on` Enables modes

`off` Disables modes

`info` Displays status information

The number of entries for each ring buffer can be specified using *size*. The ring buffer is created with a default size if *size* is not specified.

An incorrect parameter or a ? in any field outputs a list of the possible parameters.

Valid types are:

`assertmech`, `datab`, `debug`, `dispq`, `dummy`, `enf`, `hat`, `hwdebug`, `ipl`, `km`, `llio`, `locktest`, `proc`, `queue`, `sptmap`, `tpetape` or `all`.

These types enable the assertion or the debugging code for structures or functions of the same name or related structures or functions.

`dump` [`-s` *sym*] *type* [*args*]

Edit and output the contents of a log buffer (ring buffer). Such ring buffers can be created using the `assert` command.

When you specify the `-s` option, the symbolic name of the assertion structure segment can be specified using *sym* (default: `_kern_assert`).

type is the type of ring buffer to be output. Valid types are:

`enf`, `hat`, `km`, `llio`, `proc`, `queue` or `sptmap`.

These types output the related ring buffer.

Further parameters for the selective output of ring buffers for each type can be specified using *args*.

Miscellaneous commands

`call` *func* ([*parameter* [,*parameter*] ...])

Call a particular kernel function with the specified parameters. Up to six parameters can be passed and all must be numeric expressions. String constants are not allowed. The return value from the function is printed.

`cb` *newbase*

Change the default base used by the expression evaluator for interpreting numeric constants to *newbase*. (Be sure to express *newbase* in the current base.)

`echo` [*parameter* ...]

Print the parameters specified on the command line.

`ex` or `q`
Exit the debugger and continue execution of the kernel from the point where the debugger was invoked.

`lck` Display all the locks currently being held by each cpu.

`ll1` [*start*] *link value offset*, *offset* ...
Display a linked list starting from the address given in *start*; *link* indicates the offset to the pointer to the next node; *value* indicates the terminating value for the linked list. Up to five fields can be displayed for each node and *offset* lists the offset value from the start address.

`ll2` [*start*] *link value low_bound high_bound*
List a linked list starting from address given in *start*. *link* indicates the offset to the pointer to the next node. *value* indicates the terminating value for the linked list. If *low_bound* and *high_bound* are specified, then bounds checking is carried out for each node.

`ltop` Show logical to physical cpu translation. That is, dump out `ltopcpu[]`.

`more` [*value*]
Set the debug "more" facility to *value*. If *value* is zero, then "more" is turned off. Otherwise, *value* indicates the number of lines in a page.

`na` {*symbol* | *address*}
Print the numeric and symbolic values associated with the numeric expression on the command line.

`pa` Force the system to panic after exiting `ikdb`.

`ptol` Show physical to logical cpu translation. That is, dump out `ptolcpu[]`.

`ring` *contr function* [`start`]
Output the command ring buffer or the response ring buffer for a specific logical controller (*contr*). The type of ring buffer to be output is specified by the *function* parameter: "command read" (`cmdr`), "command write" (`cmdw`), "response read" (`rspr`) or "response write" (`rspw`). Output can be started from this exact element by specifying `start`.

`S` {*symbol* | *address*}[, {*symbol* | *address*}[, ...]]
Print the numeric and symbolic values associated with the numeric expressions on the command line.

`set` [*flag* {`on` | `off`}]
Turn on or off a specific internal `ikdb` flag. If no parameters are given, then the list of available flags is printed.

`val` *number*
Display the value of the numeric expression *number* in hex, decimal, unsigned decimal, and octal.

`?` Print the list of commands understood by the debugger along with a few character descriptions.

DIAGNOSTICS

The self-explanatory diagnostics indicate errors in the command line or problems encountered when executing the commands.

System Administrator's Reference Manual

ikdb(8)
RM400 only

NAME

ikdb - internal kernel debugging facility

SYNOPSIS

cmd *[parameters]*

DESCRIPTION

ikdb is an optional facility than can be built (i.e. linked) into the kernel and that runs in the kernel environment. During the initial stages of bringing up a system, the debugger should be configured into the kernel, so that it will be available to help diagnose system problems. Once the system is stable, you can deconfigure the debugger to save space in the kernel.

To configure or remove the debugger, edit the file /etc/conf/sdevice.d/ikdb:

```
ikdb Y 1 0 0 0 0 0 0
```

The second field (Y or N) is the configure parameter. If it is set to Y, the debugger will be linked into the kernel. If the field is set to N, the debugger is excluded from the kernel.

This configuration takes effect the next time you rebuild the kernel.

ikdb can be entered in several ways:

1. ikdb is called from several key points in kernel code:
   a) as early as possible in the startup process. This can be enabled by setting bootflags to 0x20.
   b) during execution of an exec call immediately prior to entering user code. This can be enabled/disabled by the on/off ikdb commands.
   c) from panic to allow for postmortem analysis. This can be disabled by setting bootflags to 0x10.
2. ikdb can be entered asynchronously by pushing the debug button or by typing CTRL-@ IKDB.
3. ikdb is entered as the result of encountering a breakpoint established by ikdb.

ikdb maintains the notion of a "current process" and a "current stack". On entry, the current process is that process on behalf of which the system is now executing (i.e. the "owner" of the current user structure). The current process can be changed to another process by the ne or cp commands so that processes that are currently dormant can be manipulated. The ne and cp commands (for multi-systems) also change the current stack to that of the new current process.

ikdb commands have the following form:

command parameters

where *command* is a command (described below) and *parameters* are parameters or options to be applied to the command. If several parameters are enclosed in square brackets ([□]), they are optional, and zero or more can be specified. If several parameters are enclosed in braces ({□}), exactly one must be specified. The output of most commands is stopped every 20 lines to allow the operator the option of terminating the command. The backspace key deletes the last character typed and CTRL-U will delete the entire command line.

Numeric expressions

Numeric expressions use a syntax compatible with the C programming language and consist of arbitrary combinations of numeric strings, kernel variables and arithmetic operators (+ - * / < > & | ^ ~). The arithmetic operators have the same meaning as they do in C. An additional operator (@) is available to specify the value found at a particular virtual address. Placing a numeric expression inside square brackets ([*expression*]) has the same effect as the indirect operator * in C.

Expressions are evaluated left to right with operator precedence as in C. In addition, parentheses can be used

to control the order of evaluation. The indirect operator can be used with additional indirect operators to achieve multiple levels of indirect accesses.

Numeric constants are evaluated as base 16 numbers unless preceded by the prefix `0b` for binary (base 2), `0` or `0o` for octal (base 8), `0t` for decimal (base 10), or `0x` for hex (base 16). The default base can also be changed with the `cb` command.

Symbolic addressing

Global kernel symbols can be used in any numeric expression to specify the address associated with the variable or procedure name. Kernel symbols are denoted by prepending the underscore character (_) to the symbol name.

COMMAND SYNTAX AND DESCRIPTION

Display/modify memory commands

`da` [{*address* | `n` | `l`} [*count*]]
`dc` [{*address* | `n` | `l`} [*count*]]
`dh` [{*address* | `n` | `l`} [*count*]]
`dl` [{*address* | `n` | `l`} [*count*]]

If *count* is negative, then memory is displayed backwards from *address*.

`ds` [{*address* | `n` | `l`} [*count*]]

Display memory contents in ascii (`da`), as disassembled code (`dc`), in hex (`dh`), as kernel symbols (`dl`), or as a string (`ds`). These are simply aliases for the `sm` command.

`ma` [{`-b` | `-h` | `-w`}] [*address* [[*value*]]
`put` [{`-b` | `-h` | `-w`}] [*address* [[*value*]]

The modifying address command is used to change the contents of the specified address. *address* is a numeric expression with the result being used as the address (either memory or its location) to be modified. The `-b` or `-h` or `-l` option specifies the typ of the value as byte or half-word or word. If numeric expression *value* is given, the result is written to address.

If no value is specified, an interactive address editor is started. It is similar to that in `mm` command but the location specified by *address* is *not* read.

`ma` and `put` are specially designed to execute the function of the `mm` command on "memory-mapped-io" addresses.

`mm` [{*address* | `n` | `l`} [*value*]]

The modify memory command is used to change the contents of the specified memory location. *address* is a numeric expression with the result being used as the address of the memory location to be modified. Specifying `n` or `l` for the address has the same meaning as for the display commands. If numeric expression *value* is given, the result is put into memory.

If no value is specified, an interactive memory editor is started. The editor will display the specified address and the contents of the memory location at that address. The value can be changed simply by entering a new value. The editor will then display the next address and contents of memory at that address. To end the process, enter a dot (".") for the value. Entering a dash ("-") for the value will decrement the address and continue the editing process. The memory editor accepts only constants; numeric expressions are not permitted.

`sa` [{`-b` | `-h` | `-w`}] [*address*]
`get` [{`-b` | `-h` | `-w`}] [*address*]

The show address command displays the contents of the specified address as byte (`-b`), word (`-w`) or long (`-l`) value in hexadecimal.

`sa` and `get` are specially designed to execute the function of the `sm` command on "memory-mapped-io" addresses.

`sm` [{`-a` | `-i` | `-l` | `-s` | `-x`}] [{*address* | `n` | `l`} [*count*]]

The show memory command displays the contents of memory in ascii (`-a`), as disassembled

instructions (`-i`), as kernel symbols (`-l`), as a string (`-s`), or in hex (`-x`). Display starts at the address specified by *address* and continues for *count* words. If *address* is not specified, the address beyond the end of the last display command is used. The start address can also be specified with respect to the last display command (`n` or `l`). Using `n` indicates that the next address should be used allowing the display to continue on from where the previous one left off. To reuse the previous start address, specify the address as `l`.

The default count of 4 is used if the parameter *count* is missing. When formatting as a string, the display will terminate whenever the count has been reached or a `NULL` character is encountered. A non existent or zero count for strings will display memory until a `NULL` is encountered.

`un` Undo the last modify memory command. A modification has a global lifetime. Hence the previous `mm` command can have occurred in a previous entry to `ikdb`. Also note that this will only undo the last change of a sequence of modifications made in the memory editor.

Display/modify context commands

`cp` *cpu*

The processor executing `ikdb` is changed to the new *cpu*. This also has the side effect of changing the current (and original) process to the one currently active on the new *cpu*. This function is only available in the multikernel.

`mtf` {`s` | `r`}

By default, this command displays a stack trace. With the `s` option, a stack trace is displayed for all sleeping processes. With the `r` option, a stack trace is displayed for all running processes.

`ne` [[`p`] *process*]

The debugger environment is changed to reflect the state of the specified process. If no processes are given, `ikdb` reverts back to its original context. *process* can be either the address of the process table entry, or if the `p` option is specified, the process ID of the new process.

`np` *cpu*

Run the debugger on *cpu*. This function is only available in the multikernel.

`regs` [`-a`] [*address*] [*reg*[=*val*]]

The `regs` command displays the contents of the processor's registers. The `-a` option displays the contents before the last panic or assertion failure. Supplying an optional *address* displays the registers in the exception frame at that location. The entire register set is displayed unless a specific register is designated by *reg*. If the numeric expression =*val* is specified, the contents of the specified register are changed to *val*.

`fregs` Display the floating point registers.

`sf` *level*

The `sf` command sets the registers and the state of `ikdb` to the specified stack frame, making it look as though execution had been interrupted at that point.

`tf` [*levels*] [*sp pc*]

This command displays a trace stack frames. For each frame on the stack, the function name, instruction address, first four parameters, and the stack frame address are printed. If *levels* is specified, the trace only proceeds for the number of stack frames specified rather than tracing back to the beginning of the stack. The trace starts from the current top of stack unless the *sp* and *pc* parameters are given.

Breakpoint and single step commands

`bp` [`-a`] *address* [*count*]

Set a breakpoint at the specified *address*. If *address* is at the beginning of a function, the breakpoint is placed at the end of the basic block instruction preamble generated by the compiler. This can be overridden using the `-a` option. Execution of the kernel stops and `ikdb` is restarted after the breakpoint is encountered *count* times. The *count* value defaults to 1.

`B` List all breakpoints currently set.

`d` *index*
: Delete the breakpoint specified by *index*, where *index* is the index associated with the breakpoint as shown in the list of breakpoints (see the `B` command).

`D`
: Delete all breakpoints currently set.

`so` [*count*]
: Similar to the `st` command except that function calls are executed as a single step.

`st` [*count*]
: Execute *count* (defaults to 1) machine instructions.

`w` [`–rw`] [`–p` *pattern*] [`–m` *mask*] [*address*]
: Without parameters this command shows the watchpoint currently set. With the `–rw` option a watchpoint can be specified at address for reading (`–r`), writing (`–w`) or both. Address must be from K0SEG to K1SEG.

  The watchpoint is pattern sensitive for each bit which is set to 1 by the `–m` option. It is then compared to the appropriate bit in the pattern specified by the `–p` option.

`W`
: Delete watchpoint currently set.

Process display/control commands

`fp` {`p` | `c` | `w` | `s`} *regexp*
: Find and list all processes that match a specific selection criterion. Exactly one flag must be given to indicate the field for the search: process ID (`p`), command (`c`), wait channel (`w`), or process state (`s`). *regexp* is a regular expression used to compare against the given field. For example:

  ```
  fp c ^get
  ```

  would find and list all processes executing commands with names beginning with "get".

`ki` [`–s`] [*–sig*] [*pid*]
: Send a signal to a process. If specified, *sig* specifies the signal to send; 9 is the default. *pid* indicates the process to signal. If *pid* is missing, then all processes except system processes are signaled. System processes cannot be killed unless the `–s` option is specified and the process ID is stated explicitly.

`lp`
: The `lp` command lists all the processes active in the system and gives a short status description. The processes that are currently running on a CPU are highlighted.

`wa` *wchan*
: Perform the Reliant UNIX wakeup call on the specified wait channel. *wchan* must be specified; there are no defaults.

Data structure display commands

This group of commands prints the contents of data structures. For most of these commands, the structure can be specified by the following arguments:

{*address* | `offset`}

If *address* is given, the command prints the name of each member of the structure (truncated to 8 characters), and the member's value, as in the following example for `percpu`:

```
ikdb > percpu c0030000□
c0030000:□
xclbind         0 idleproc c03aa400 lcpunum         1 lcpumask        2 □
percpufl        7 runrun           1 kprunrun        0 curpri         4f □
npwakecn        0 dosoftin         1 dotimein        0 dqgen           1 □
migrate_        0 intrcxt        100 locktestoff     0 lastlock       ff □
locklist [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0] □
idleflag        0 backoff          5 idleadj         0 baseprio        0 □
fp_proc         0 curproc  c0739c00 old_curp c03aa400 tlbpid_a       29 □
tlbpid_g       ca curipl           0 maxqueip        0 iplquema e0118000 □
```

```
iplfree  c00479ec iplbase  c0004000 unix-tic        1 fast_tod        0□
fast_off        0 Izero           0 itmr_ckp        0 rtod_ckp        0□
tod_hi_d
       0 clk_spee        0 delay_mu      708 icache_s        0□
dcache_s        0
```

If `offset` is specified, the printout specifies the full name of each structure member, its offset from the beginning of the structure in bytes, and its size in bytes. The following is an example of the `percpu` command:

```
ikdb > percpu offset□
member(offset,size)□
xclbind(0,4)         idleproc(4,4)        lcpunum(8,1)         lcpumask(c,4) □
percpuflags(10,4)    runrun(20,4)         kprunrun(24,4)       curpri(28,4) □
npwakecnt(2c,4)      dosoftint(30,4)      dotimein(34,4)       dqgen(38,4) □
migrate_from(3c,4)   intrcxt(60,2)        locktestoff(62,1)    lastlock(63,1) □
locklist(64,7c)      idleflag(e0,4)       backoff(e4,4)        idleadj(f0,4) □
tlbpid_assigned(12c,4) tlbpid_gen(130,4) curipl(134,4)        maxqueipl(138,4) □
iplquemask(13c,4)    iplfree(140,4)       iplbase(144,4)       unix_tick(180,4)□
fast_tod_lo(184,4)   tick_tod_hi(188,4)   tick_tod_lo(184,4)   show_tod_hi(190,4)□
show_tod_lo(194,4)   fast_offset(198,4)   Izero(190,4)         itmr_ckpt_cnt(1a0,4)□
rtod_ckpt_cnt(1a4,4)
tod_hi_diff(1a8,4) clk_speed(1a4,1)    delay_mult(1b0,4)□
icache_size(1b4,4)   dcache_size(1b8,4)
```

`as` {*address* | `offset`}
: Displays the contents of a particular `as` structure.

`buf` {*address* | `offset`}
: Displays the contents of a particular `buf` structure.

`cg` {*address* | `offset`}
: Displays the contents of a particular `cg` structure.

`datab` {*address* | `offset`}
: Displays the contents of a particular `datab` structure.

`fifonode` {*address* | `offset`}
: Displays the contents of a particular `fifonode` structure.

`hat` {*address* | `offset`}
: Displays the contents of a particular `hat` structure.

`hat_data` {*address* | `offset`}
: Displays the contents of a particular `hat_data` structure.

`inode` {*address* | `offset`}
: Displays the contents of a particular `inode` structure.

`inpcb` {*address* | `offset`}
: Displays the contents of a particular `inpcb` structure.

`msgb` {*address* | `offset`}
: Displays the contents of a particular `msgb` structure.

`page` {*address* | `offset`}
: Displays the contents of a particular `page` structure.

`percpu` {{*lcpu* | *address*} | `offset`}
: Displays the contents of the `percpu` structure for a particular processor. The processor is specified by either its logical CPU number (*lcpu*) or the address of its `percpu` structure (*address*). If `offset` is given, then the offset of each member into the structure is printed as described above.

  If no arguments are given, this command prints the values for the current active CPU's `percpu` structure (note the active `lcpu` number in the prompt) with the structure member names truncated to 8

characters.

`pp` {{*pid* | *address*} | `offset`}
: Displays the contents (in logical order) of the *proc* structure for a particular process. The process is specified by either its process ID (*pid*) or the address of its process table entry (*address*).

`ppa` {{*pid* | *address*} | `offset`}
: Displays the contents (in alphabetical order) of the *proc* structure for a particular process. The process is specified by either its process ID (*pid*) or the address of its process table entry (*address*). If `offset` is specified, then the offset into the structure of each field is printed.

`queue` {*address* | `offset`}
: Displays the contents of a particular `queue` structure.

`seg` {*address* | `offset`}
: Displays the contents of a particular `seg` structure.

`snode` {*address* | `offset`}
: Displays the contents of a particular `snode` structure.

`stream` {*address* | `offset`}
: Displays the contents of a particular `stream` structure.

`tcpcb` {*address* | `offset`}
: Displays the contents of a particular `tcpcb` structure.

`var` [`offset`]
: Displays the contents of the `var` structure (v).

`vnode` {*address* | `offset`}
: Displays the contents of a particular `vnode` structure.

Virtual memory translation commands

`pt` *address*
: Display the translation information associated with the given virtual address. This includes the physical address, and protection flags from both the kernel's data structures and the hardware TLB (Translation Lookaside Buffer).

`td` [`-c`] [*index* ...]
: Display the contents of the TLB. When `ikdb` is started, the contents of the TLB are saved and used for this command. To use the current contents of the TLB, specify the `-c` option. To dump specific TLB locations, specify the indices of the locations to be dumped.

`vtop` *proc vaddr*
: Display the steps of the virtual to physical address translation for address pair *proc vaddr*.

Miscellaneous commands

`call` *func* ([*parameter* [,*parameter*] ...])
: Call a particular kernel function with the specified parameters. Up to six parameters can be passed and all must be numeric expressions. String constants are not allowed. The return value from the function is printed.

`cb` *newbase*
: Change the default base used by the expression evaluator for interpreting numeric constants to *newbase*. (Be sure to express *newbase* in the current base.)

`echo` [*parameter* ...]
: Print the parameters specified on the command line.

`ex` or `q`
: Exit the debugger and continue execution of the kernel from the point where the debugger was invoked.

`lck` Display all the locks currently being held by each cpu.

`ll1` [*start*] *link value offset*, *offset* ...
: Display a linked list starting from the address given in *start*; *link* indicates the offset to the pointer to the next node; *value* indicates the terminating value for the linked list. Up to five fields can be displayed for each node and *offset* lists the offset value from the start address.

`ll2` [*start*] *link value low_bound high_bound*
: List a linked list starting from address given in *start*. *link* indicates the offset to the pointer to the next node. *value* indicates the terminating value for the linked list. If *low_bound* and *high_bound* are specified, then bounds checking is carried out for each node.

`more` [*value*]
: Set the debug "more" facility to *value*. If *value* is zero, then "more" is turned off. Otherwise, *value* indicates the number of lines in a page.

`na` {*symbol* | *address*}
: Print the numeric and symbolic values associated with the numeric expression on the command line.

`pa`
: Force the system to panic after exiting `ikdb`.

`S` {*symbol* | *address*}
: Print the numeric and symbolic values associated with the numeric expression on the command line.

`set` [*flag* {`on` | `off`}]
: Turn on or off a specific internal `ikdb` flag. If no parameters are given, then the list of available flags is printed.

`val` *number*
: Display the value of the numeric expression *number* in hex, decimal, unsigned decimal, and octal.

`?`
: Print the list of commands understood by the debugger along with a few character descriptions.

DIAGNOSTICS

The self-explanatory diagnostics indicate errors in the command line or problems encountered when executing the commands.

System Administrator's Reference Manual

inst_dbg_kernel(8)
RM600 only

NAME

`inst_dbg_kernel` - set up debug kernel environment

SYNOPSIS

`/usr/sbin/inst_dbg_kernel` [`–m` *mountpoint*] [`–r` *respdir*]

`/usr/sbin/inst_dbg_kernel –h`

DESCRIPTION

This command installs the packages of the debug kernel under `/.utsdbg/etc/conf` and copies additional files from `/etc/conf` in order to set up an environment for generating a debug kernel.

To do this, the correct Reliant Unix CD for the installed system must be already mounted; this CD contains the packages for the debug kernel. The mount point can be specified with the `–m` *mountpoint* option.

The `inst_dbg_kernel` command installs the debug kernel packages, for example `utsdbg` and `adddbg`, from the Reliant UNIX CD using `pkgadd(1M)` .

If single correctional releases containing kernel components are installed on your system, these debug kernel components must be installed following successful execution of `inst_dbg_kernel` in accordance with the correctional releases (see **DIAGNOSTICS**).

Please refer to the `utsdbg(5)` manual page for an overview of the installation procedure, generation, as well as the activation and deletion of the debug kernel.

OPTIONS

`–m` *mountpoint*

`inst_dbg_kernel` uses the *mountpoint*, where the directory structure of the system CD is expected to be mounted.

The standard value for *mountpoint* if the `–m` option is not specified is `/cdrom`.

`–r` *respdir*

When the `–r\f option is specified, inst_dbg_kernel` uses response files, which are expected to be found in the *respdir* directory.

If the `–r` option is not specified, response files are not used. If the debug kernel packages contain request files, the queries must be answered interactively during the installation.

To date, however, request files have not been provided for the debug kernel packages.

`–h` An explanatory syntax message is displayed.

DIAGNOSTICS

Since `inst_dbg_kernel` invokes other commands to create the environment for generating the debug kernel, errors detected with this command are also reported. In general terms, the return code 1 or 10 indicates that `inst_dbg_kernel` has detected an error.

The return code 10 indicates a syntax error when invoking `inst_dbg_kernel`.

The return code 1 is used for other errors that have occurred.

All error messages should be comprehensible without further explanations.

If `inst_dbg_kernel` issues a warning that single correctional releases are installed on your system, check if these releases contain kernel components. If they do, you need to get the corresponding debug kernel components and install them as well. Otherwise, your debug kernel will not contain the error recoveries for the correctional releases that affect the kernel.

FILES

The package names installed with `inst_dbg_kernel` (value of the `PKG` package variables) are recorded in a file and used for information purposes when deleting using `del_dbg_kernel(8)`.

SEE ALSO

`pkgadd(1M)`, `utsdbg(5)`, `del_dbg_kernel(8)`, `setup_dbg_kernel(8)`.

System Administrator's Reference Manual

iosreprobe(8)

NAME

`iosreprobe` - reprobe a SCSI port

SYNOPSIS

`/sbin/iosreprobe` *portname*

DESCRIPTION

The `iosreprobe` utility performs an update of the `autoconf` structures of the specified port.

Its only purpose is to attach devices to the system, which were powered down at boot-time, and are now to be turned on again. In most cases these devices will be line printers, external tape drives or similar.

`iosreprobe` must not be used to detach or change devices.

portname is the SCSI port name as it is displayed with `autoconf(8)`.

SEE ALSO

`devname(7)`, `autoconf(8)`, `iosrestart(8)`.

System Administrator's Reference Manual

iosrestart(8)

NAME

`iosrestart` - restart a SCSI device, port, or controller

SYNOPSIS

`/sbin/iosrestart` *devname*

`/sbin/iosrestart` *portname*

`/sbin/iosrestart` *ctlrname*

DESCRIPTION

The `iosrestart` utility performs a reinitialization of SCSI software and hardware. `iosrestart` is an interactive utility. The user receives prompts through each step of the process.

devname is the SCSI device name as it is displayed with `autoconf(8)`.

portname is the SCSI port name or bus node as it is displayed with `autoconf(8)`.

ctlrname is the name of the SCSI controller.

`iosrestart` *devname*

Restart a single device. This command performs a hard reset of the specified device. The utility probes the specified device to ensure that it still exists and then reinitializes all the software structures associated with that device.

This command has no effect on any other device, on the port, or on the controller.

`iosrestart` *portname*

Restart a chain of devices on the same SCSI port. This command performs a hard reset of the specified SCSI port (bus) which results in a hard reset of each device attached to that port. After the reset, the specified SCSI port is probed. All of the software structures associated with each device on that port are reinitialized.

This command has no effect on the controller node or any device attached to a neighboring port on the same controller.

`iosrestart` *ctlrname*

Reinitialize the SCSI controller card and all attached devices. This command performs a hard reset of the controller which means a hard reset of all the devices attached to that controller. After the reset, the controller is probed to verify that the controller and its devices exist. All of the software structures associated with that controller and its devices are reinitialized.

NOTES

`iosrestart` cannot be performed on the root device, its port, or its controller.

`iosrestart` cannot be performed on a device that is configured as a system swap device.

SEE ALSO

`devname(7)`, `autoconf(8)`.

System Administrator's Reference Manual

iossetup(8)

NAME

`iossetup -` set up and configure SCSI devices

SYNOPSIS

`/sbin/iossetup` *setup_parameter*

DESCRIPTION

`iossetup` can be used instead of the individual setup commands listed under **SEE ALSO**, namely `dksetup`, `fdsetup` etc.

setup_parameter offers the same options and parameters as with the specific setup commands. For a description of their meaning, please refer to the individual commands descriptions.

The *devname* parameter of the special setup command forces the `iossetup` command to branch without modification to the special setup command. The plausibility checks are carried out here, and any error messages are output by the special setup command.

EXAMPLE

To update a specific device on an IOS SCSI disk, you can invoke

```
iossetup -U /dev/ios0/rsdisk012s0
```

instead of

```
dksetup -U /dev/ios0/rsdisk012s0
```

and achieve the same result.

DIAGNOSTICS

Error messages and return values correspond to those of the special setup command.

SEE ALSO

`dksetup`(8), `fdsetup`(8), `jbsetup`(8), `sgensetup`(8), `tpsetup`(8).

System Administrator's Reference Manual

jbsetup(8)

NAME

`jbsetup -` initialize/update physical jukebox configuration

SYNOPSIS

`jbsetup –U` *devname*

DESCRIPTION

`jbsetup` initializes a new jukebox for use with the system software. Jukebox administration performed by `jbsetup` consists of building the correct device node in `/dev`.

devname is the generic name of a jukebox device as displayed by `autoconf`. Refer to `sjuke(7)` for more information.

`–U` upgrades a jukebox for the current release of the system software. This is non-destructive.

SEE ALSO

`devname(7)`, `sjuke(7)`, `autoconf(8)`.

System Administrator's Reference Manual

ldcontroller(8)
RM600 only

NAME

`ldcontroller -` load information from /etc/controller into NVRAM

SYNOPSIS

`/sbin/ldcontroller [ –c ] [ –f` *file* `]`

DESCRIPTION

`ldcontroller` reads the `/etc/controller` file, checks its contents for correct syntax and consistency and writes the controller configuration to NVRAM. The NVRAM is only modified if `ldcontroller` established no errors beforehand.

OPTIONS

`–c` The file `/etc/controller` is merely checked. The NVRAM remains unchanged.

`–f` *file*
The specified file *file* is used instead of `/etc/controller`.

DIAGNOSTICS

`ldcontroller` ends with an exit code other than 0 in the event of errors. The error messages are self-explanatory.

SEE ALSO

`controller(4)`, `mkcontroller(8)`, `modcontroller(8)`, `showconf(8)`.

mkcontroller(8) - wsinit(8)

System Administrator's Reference Manual

mkcontroller(8)
RM600 only

NAME

`mkcontroller -` generate standard data for /etc/controller

SYNOPSIS

`/sbin/mkcontroller –a|–i`

DESCRIPTION

`mkcontroller` creates a standard version of the `/etc/controller` file on standard output. `mkcontroller` can be used by the system administrator in order to reconfigure the system with the default values, or to clarify differences from the standard configuration.

OPTIONS

`–a` A version of `/etc/controller` is written to standard output, where `/etc/controller` has been extended with new controller entries.

`–i` A version of `/etc/controller` is displayed on standard output. This version corresponds to the version of `/etc/controller` stored in the NVRAM and contains additional entries for new controllers (not defined in the NVRAM). Entries relating to non-existent controllers are not transferred. If the NVRAM is not initialized, then all controllers are assumed to be new.

DIAGNOSTICS

`0` = No errors

`1` = NVRAM not initialized

`2` = Error

SEE ALSO

`controller(4)`, `ldcontroller(8)`, `modcontroller(8)`, `showconf(8)`.

System Administrator's Reference Manual

modcontroller(8)
RM600 only

NAME

`modcontroller` - modify /etc/controller

SYNOPSIS

`/sbin/modcontroller`

DESCRIPTION

`modcontroller` enables the `/etc/controller` file to be modified safely. For the actual changes, `modcontroller` calls up an editor, which works on a copy of `/etc/controller`. The editor defined in the environment variable `$EDITOR` is called, and the standard editor is "vi". `modcontroller` checks the modifications using `ldcontroller(8)`. `/etc/controller` is not modified while `ldcontroller` is showing errors.

SEE ALSO

`controller(4)`, `ldcontroller(8)`, `mkcontroller(8)`, `showconf(8)`.

System Administrator's Reference Manual

nvram(8)
RM400 only

NAME

`nvram -` print and modify nonvolatile memory data

SYNOPSIS

`/sbin/nvram` [`-fsa`] [`-b` *bootfile*] [`-B`] [`-S` *bootmode*] [`-i` *inet_addr*] [`-t` [*args*]] [`-c` [*args*]] [`-0|1` *args*] [`-n c|e`] [`-x` [*string*]] [`-T` *flags*] [*device*]

DESCRIPTION

`nvram` prints and modifies nonvolatile memory data. It may be used, for example, to examine (and even to patch) configuration data. `nvram` also has some additional features that relate to teleservice requirements ("shadow console").

OPTIONS

`-f` Reads firmware parameters and prints them to screen. This function prints: 1) Internet address; 2) console parameter: console type [MEM, SIO, GRAF], baudrate, v24 signals; 3) tty0; 4) tty1; 5) teleservice parameter: baudrate, v24 signals, tty setups; 6) graphic controller type; 7) graphic resolution; 8) graphic monitor; 9) network type.

`-s` Reads boot parameters and prints them to screen. This option is set by default. This function prints: bootmode, power on status, power on failcode, power on mask, reset PC, reset RA and memory parity switch. The parameters that are relevant for a remote installation are also output: the IP address of the server, the server name, the IP address of the gateway, the subnet network mask, and the installation type.

`-a` See `-f` and `-s`.

`-b` *bootfile*
Creates a bootfile entry in NVRAM. This entry is evaluated by the firmware if the system is started in boot mode "c" or manually with "auto". The string for *bootfile* can contain a maximum of 20 characters.

Example: `nvram -b "dkncr(,,10)sash"`

`-B` Outputs the *bootfile* string stored in NVRAM.

Example: `nvram -B`

The *bootfile* string stored in NVRAM is sent to stdout, e.g. `Current bootfile: dkncr(0,0,10)sash`.

`-S` Sets the bootmode in the NVRAM. This entry is evaluated by the firmware to determine the manner in which the system should power up after a reboot. The valid boot modes can be found in `prom(8)`.

Example: `nvram -S r`

`-i` *inet_addr*
Sets the internet number in the NVRAM.

`-t` Displays the current status of the Tele-X-Link modem connection.

Output: `active` or `inactive`.

`-t` *args*
Sets teleservice parameter. The following can be entered for *args*:

baudrate
`:300:600:1200:2400:4800:9600:19200:38400:`

bits per char
`:cs7:cs8:`

stopbits

```
:stop1:stop2:
```

parity

```
:parno:parodd:parevn:parinv:
```

teleservice flags

```
:tsa:tse:
```

v24 signals

```
:s1:s2:s4:m1:m2:m3:m4:m5:
```

Type ~ or ! for disabling teleservice flags or teleservice v24 signals.

Example: `nvram –t 19200:cs8:~stop1:tsa:!tse:s1:~m5 –a`

`–c` *args*

Sets console flags. The following can be entered for *args*:

baudrate

```
:75:110:134:150:300:600:1200:1800:2400:4800:9600:19200:38400:
```

v24 signals

```
:s1:s2:m1:m5:
```

Type ~ or ! for disabling console v24 signals.

Example: `nvram –c 19200:s1:!s2:~m1 –a`

`–0` *args*

`–1` *args*

Sets tty0 or tty1 parameter. The following can be entered for *args*:

v24 signals

```
:s1:s2:m1:m2:m3:m5:
```

`–n c|e`

Switches between Cheapernet (`c`) and Ethernet (`e`).

`–x` *string*

Sets an initialization string for BA80.

string not specified: The current initialization string is displayed.

`–T` *flags*

Switches the console to teleservice and back.

The following can be entered for *flags*:

`on` Connects to the teleservice port, switches to the teleservice console ("shadow console").

`off` Switches back from the teleservice console to the local console and disconnects.

`local`

Just switches console back to the local console but does not disconnect.

`status`

Displays the current status of the Tele-X-Link port.

Output:

```
Modem status : on|off
Remote Console Connection : on|off
```

device

Optional parameter. *device* is set by default to `/dev/nvram`.

NOTES

For creating of `/dev/nvram` type: `mknod nvram c 129 10`.

Ensure your Reliant UNIX version recognizes a device driver called `nvram`, otherwise your system may crash.

FILES

`/dev/nvram`

SEE ALSO

nvram(7), prom(8).

System Administrator's Reference Manual

odr(8)

NAME

`odr` - Online Drive Replacement (only applies for OS3x/OS4x jukebox)

SYNOPSIS

`odr –l` *sgen_device-node*

`odr –e [ –b ]` *sdisk_device-node sgen_device-node*

`odr –d [ –b ]` *sdisk_device-node sgen_device-node*

`odr –e –u` *x sgen_device-node*

`odr –d –u` *x sgen_device-node*

DESCRIPTION

`odr` supports the replacement of drives within an OS3x/OS4x jukebox. It is a component of the `SIsgen` package. The command was created for trained technicians, to allow them to perform diagnostics on the drives in a jukebox and to replace them. The command can be used to query the status of the drives in a jukebox, to identify a particular drive as being defective, or to start up a drive again after it has been replaced. In all cases, either the "autoconf" name, or the path name of the designated jukebox must be specified. The individual functions are controlled by means of the options described below:

OPTIONS

`–l` Displays the status of all drives in the jukebox, as well as the allocation of the logical drive number to the SCSI ID used.

Possible values for the drive status are as follows:

"Drive not valid", "Online Good", "Online Pending", "Offline Failed", "Offline Good Pending" or "Offline Good".

The "Online Good" status indicates a drive that is ready to operate.

The "Online Pending" status means that the drive was defective and has meanwhile been replaced.

Depending on the type of drive installed, it is also possible to have dual drives in the jukebox. While these drives are not defective they will be replaced as well. The "Offline Good Pending" status is activated automatically with these dual drives, when the technician closes down the second drive affected. Once the drive has been replaced, the status of the second drive switches to "Offline Good".

`–d` Deactivates a drive in the jukebox. The drive can either be specified with the "autoconf" name (e.g. `ios0/sdisk044`) or with the logical number within the jukebox. If the "autoconf" name is specified, the device node identified is deactivated for the device driver.

`–e` Activates a drive in the jukebox. The drive can either be specified with the "autoconf" name (e.g. `ios0/sdisk044`) or with the logical number within the jukebox. If the "autoconf" name is specified, the device node identified is reactivated for the device driver.

`–b` This option must be used if the `sdisk` device node is located on a different SCSI channel than the `sgen` device node.

`–u` *x* If only the logical number is known within the jukebox, this option must be used instead of the `sdisk` "autoconf" name. This means, for example, that `–u 3` must be specified for the logical drive 3. Since the "autoconf" name is not known in this case, no changes are made at the device node.

EXAMPLES

Example of a replacement sequence:

In order to analyze the status of the drives in a `ios0/sgen036` jukebox, the following command must be used:

```
# odr -l ios0/sgen036
```

This command returns the following:

```
Drive  1, Bus: 1, SCSI-ID: 5, Status: Online Good
Drive  2, Bus: 1, SCSI-ID: 4, Status: Online Good
Drive  3, Bus: 1, SCSI-ID: 3, Status: Online Good
Drive  4, Bus: 1, SCSI-ID: 2, Status: Online Good
Drive  5, Bus: 1, SCSI-ID: 1, Status: Online Good
Drive  6, Bus: 1, SCSI-ID: 0, Status: Online Good
```

Assuming `ios0/sdisk033` has reported an error and is to be deactivated, the following command must be used:

```
# odr -d ios0/sdisk033 ios0/sgen036

Once deactivated, the situation looks as follows:# odr -l ios0/sgen036
Drive  1, Bus: 1, SCSI-ID: 5, Status: Online Good
Drive  2, Bus: 1, SCSI-ID: 4, Status: Online Good
Drive  3, Bus: 1, SCSI-ID: 3, Status: Offline Failed
Drive  4, Bus: 1, SCSI-ID: 2, Status: Online Good
Drive  5, Bus: 1, SCSI-ID: 1, Status: Online Good
Drive  6, Bus: 1, SCSI-ID: 0, Status: Online Good
```

If the technician now replaces `ios0/sdisk033`, `ios0/sdisk032` is also affected because the two form a dual drive. The technician now places a drive in the jukebox that is no longer accessible, even to `ios0/sdisk032`. The jukebox now assumes the following status:

```
# odr -l ios0/sgen036
Drive  1, Bus: 1, SCSI-ID: 5, Status: Online Good
Drive  2, Bus: 1, SCSI-ID: 4, Status: Online Good
Drive  3, Bus: 1, SCSI-ID: 3, Status: Offline Failed
Drive  4, Bus: 1, SCSI-ID: 2, Status: Offline Good Pending
Drive  5, Bus: 1, SCSI-ID: 1, Status: Online Good
Drive  6, Bus: 1, SCSI-ID: 0, Status: Online Good
```

Further access from the host to `ios0/sdisk032` can be prevented using the following command:

```
# odr -d ios0/sdisk032 ios0/sgen036
```

The output returned by

```
# odr -l ios0/sgen036
```

is as follows:

```
Drive  1, Bus: 1, SCSI-ID: 5, Status: Online Good
Drive  2, Bus: 1, SCSI-ID: 4, Status: Online Good
Drive  3, Bus: 1, SCSI-ID: 3, Status: Offline Failed
Drive  4, Bus: 1, SCSI-ID: 2, Status: Offline Good
Drive  5, Bus: 1, SCSI-ID: 1, Status: Online Good
Drive  6, Bus: 1, SCSI-ID: 0, Status: Online Good
```

After the technician has replaced the dual drive, the status is as follows:

```
# odr -l ios0/sgen036
Drive  1, Bus: 1, SCSI-ID: 5, Status: Online Good
Drive  2, Bus: 1, SCSI-ID: 4, Status: Online Good
Drive  3, Bus: 1, SCSI-ID: 3, Status: Online Pending
Drive  4, Bus: 1, SCSI-ID: 2, Status: Online Pending
Drive  5, Bus: 1, SCSI-ID: 1, Status: Online Good
Drive  6, Bus: 1, SCSI-ID: 0, Status: Online Good
```

The commands:

```
# odr -e ios0/sdisk032 ios0/sgen036
# odr -e ios0/sdisk033 ios0/sgen036
```

are used to reactivate the specific drives in the jukebox, and reopen access from the host. The startup status is then active again:

```
# odr -l ios0/sgen036
Drive  1, Bus: 1, SCSI-ID: 5, Status: Online Good
Drive  2, Bus: 1, SCSI-ID: 4, Status: Online Good
Drive  3, Bus: 1, SCSI-ID: 3, Status: Online Good
Drive  4, Bus: 1, SCSI-ID: 2, Status: Online Good
Drive  5, Bus: 1, SCSI-ID: 1, Status: Online Good
Drive  6, Bus: 1, SCSI-ID: 0, Status: Online Good
```

If the affected `sdisk` is located on a different SCSI port than the `sgen` node (such as `ios0/sdisk054`), the `–b` option must be added. The deactivate commands then looks as follows:

```
# odr -d -b ios0/sdisk054 ios0/sgen036
```

RETURN VALUES

If the `odr` command has been executed successfully, it returns a 0. If this is not the case, a corresponding error message is output and a value other than 0 is returned.

SEE ALSO

`sdisk`(7), `sgen`(7), `autoconf`(8).

System Administrator's Reference Manual

preload(8)

NAME

`preload -` preload and lock text into memory

SYNOPSIS

`/usr/bin/preload` [*filename*]

`/usr/bin/preload –d` [*filename*]

`/usr/bin/preload –l`

DESCRIPTION

`preload` is used to preload the text segment of binary objects linked with `–B ptc_bigpage` option. Preloading a file into memory ensures that enough contiguous physical memory pages are allocated for the whole text segment, prior to executing other application files. Files that are preloaded into memory are locked, and will not be paged out of memory. System performance is improved in two ways:

1. There is only one TLB (Translation Lookaside Buffer) entry used to map the whole text segment. This TLB is locked to eliminate text TLB misses and consequently increase TLB hits for other memory segment.
2. The text pages are locked in the memory and thus there are no paging or swapping activities on the text pages.

The largest possible text segment that can be preloaded is 32 MB. Only one file can be preloaded at the present time.

OPTIONS

`–d` *filename*
: Unload preloaded files. Preloaded files can also be unloaded by unmounting the file system containing the preloaded file. Both operations will fail if there are active processes executing that file.

`–l`
: List preloaded files.

DIAGNOSTICS

`preload` returns exit code 0 if the file was preloaded or unloaded successfully. Otherwise, it prints a diagnostic message and returns non-zero.

SEE ALSO

`exec(2)` .

System Administrator's Reference Manual

prom(8)
RM400 only

NAME

`prom` - PROM monitor used to perform power-on tests and tobootstrap the SASH

SYNOPSIS

The PROM monitor resides in PROM as firmware and is invoked automatically via power-on or reset.

DESCRIPTION

The PROM monitor is used to perform the startup procedure after power-on (power-on tests and basic initializations) and to boot the SASH (stand-alone shell). The commands provided by the PROM are the same as the SASH commands, apart from the `cp` command. The PROM monitor is entered after the power-on tests have been run. The next actions performed by the PROM monitor depend on the `bootmode` environment variable; this variable is stored in non-volatile RAM and retains its value during power-off.

Bootmode	Action
_	
d	Power-on tests are skipped. Memory is cleared. The PROM monitor waits for user input.
m	Power-on tests are performed. If an error occurs, the bootmode is changed automatically to "e" (error). After completing the tests, the PROM monitor waits for user input.
c	Power-on tests are performed. If an error occurs, the bootmode is changed automatically to "e" (error). If no error occurs, the PROM monitor automatically boots the SASH (which will subsequently boot the Reliant UNIX operating system automatically).
e	An error occurred in a previous run of the power-on tests. This bootmode will not be reset automatically to another value ("m" or "c") if the next run of the power-on tests is error free.
w	Warm start is performed at reset
r	A remote installation is started from the firmware (after a reset or init 6). The boot mode is subsequently set again to "c".
i	An installation is started from the system CD (after a reset or init 6). The boot mode is subsequently set again to "c".

The name of the file to boot automatically when the `bootmode` environment variable has the value "c" is stored in the `bootfile` environment variable. These environment variables can be changed with the PROM monitor command `setenv` and displayed with the `printenv` command. They both retain their values during power-off as they are stored in non-volatile RAM.

When waiting for user input, the PROM monitor issues the prompt `>>`.

At almost any time, pressing ^D or ^C will cause the PROM monitor to cancel any commands issued and fall back into its command parser to await user input again.

SEE ALSO

`nvram`(7), `sash`(8).

System Administrator's Reference Manual

pft(8)
RM400 only

NAME

`ptf` - command for checking the voltage and temperature sensors

SYNOPSIS

`/sbin/ptf [-p][-t][-f][-b][-P][-T][-F][-B]`

DESCRIPTION

The `ptf` command allows the user to query the current status of the BBU, the voltage, temperature, and fan sensors for using the `/dev/at/ptf` device [see `ptf(7)`].

A maximum of 8 devices of the same type are supported in a system cabinet (e.g. up to 8 power supply units). If there is no device of the requested type available in the system cabinet, "–" is output. The output is left justified. The devices are output in the following sequence: no0, no1, no2, no3, no4, no5, no6, no7.

OPTIONS

`-p` Displays the status of the power supply unit – A letter is output for every device identified.

Either a "P" or "p" is output for every available power supply unit that is identified. A "P" is output if a power supply unit is identified as being available, but neither an environment error (power failure) nor a device error (defective power supply unit) was detected. A "p" means that a device is available but an error has occurred.

Example:

```
# ptf -p
Pp
```

Explanation:

Two power supply units were identified, one of which (no0) is Ok. The other power supply unit (no1) has reported an error. (A "PP" cannot arise at present because a redundant power supply unit cannot be identified in the normal state)

`-P` Displays the status of the power supply unit – A hexadecimal value is output for every device identified.

Each device status is composed of four bits (therefore in total a Nibble).

`0x1` Device is identified (exists)

`0x2` Device error (e.g. defective power supply unit)

`0x4` Special bit (reserved, but not used at present)

`0x8` Environment error (e.g. power failure)

Example:

```
# ptf -P
19000000
```

Explanation:

Two power supply units were identified, one of which (no0) is Ok. The other power supply unit (no1) has reported an environment error (power failure).

`-t` Displays the status of the temperature sensor – A letter is output for every device identified.

Similar to `-p`. A "T" or "t" (sensor damaged or tempfail) is output for every available temperature sensor that is identified.

`-T` Displays the status of the temperature sensor – A hexadecimal value is output for every device identified.

Similar to `-P`.

`-f` Displays the status of the fan sensor – A letter is output for every device identified.

Similar to `-p`. An "F" or "f" (fanfail) is output for every available fan that is identified.

`-F` Displays the status of the fan sensor – A hexadecimal value is output for every device identified.

Similar to `-P`.

`-b` Displays the status of the BBU – A letter is output for every device identified.

Similar to `-p`. Either a "B" or "b" (BBU damaged oder BBU active) is output for every available BBU that is identified.

Example:

```
# ptf -b □
-
```

`-B` Displays the status of the BBU – A hexadecimal value is output for every device identified.

Similar to `-P`.

Example:

```
# ptf -B □
00000000
```

FILES

`/dev/at/ptf`

NOTE

Not valid for RM400-xxx.

SEE ALSO

`ptf`(7), `ptfd`(8).

System Administrator's Reference Manual

ptfd(8)
RM400 only

NAME

`ptfd -` monitor system cabinet

SYNOPSIS

`/usr/sbin/ptfd`

DESCRIPTION

The `ptfd` daemon monitors the system cabinet and is controlled by means of the `/var/adm/ubft/_upstab` file [see `_upstab`(4)]. It reacts, in particular, to the following events:

error ::= {`acf2` | `fan_unit_fail` | `temp_fail` | `acf1` | `eip_lost` | `too_few_ps_units`}

NOTE

Not valid for RM400-xxx.

SEE ALSO

`_upstab`(4), `eipd`(8), `ptf`(8), `pupsd`(8).

System Administrator's Reference Manual

pupsd(8)
RM400 only

NAME

`pupsd` - control UPS using the protocol interface

SYNOPSIS

`/usr/sbin/pupsd`

DESCRIPTION

The `pupsd` daemon monitors the UPS (Uninterruptible Power Supply) over a V24 line using a protocol. The protocol comprises ASCII strings, which are sent to UPS and to which UPS can respond depending on the command.

The only protocol currently supported is the SIEMENS UPS 9071 (P□model).

The system needs to be connected to the UPS by means of a special cable in order for communication to take place between the `pupsd` daemon and UPS. A `/dev/term/UPS` device file must also be installed, which handles the entire communication process.

The device file is created with the `termadd(1M)` command.

Example:

```
termadd –T serial –b motherboard0,6 –n UPS
```

The `pupsd` daemon is controlled by means of the `/var/adm/ubft/_upstab` file [see `_upstab(4)`]. It reacts, in particular, to the following events:

error ::= {`ups_active` | `ups_active+batlow`}

These events have the following meaning:

`ups_active`
: Mains power supply has failed, batteries are supplying power.

`ups_active+batlow`
: The battery power is also low now.

Other important entries for a 9071 type UPS include:

`ups_polltime:`*nn*
: This parameter determines the time period (in seconds) to be waited between two polls to the UPS. The default is for the system to poll the UPS every 16 seconds for its status; this corresponds to `ups_polltime:16`.

`ups_downdelay:`*nn*
: Before the system halts, it issues a final command to the UPS informing it to disable power supply from the batteries (e.□g. to maintain reserves in case another mains failure should occur). The UPS delays the actual disable for a period of *nn* minutes.

NOTES

In order to influence the behavior of the system during power down, you have to use the `pwr` command in the shell script [see "script" in `_upstab(4)`].

SEE ALSO

`termadd(1M)`, `ups(1M)`, `_upstab(4)`, `pwr(8)`.

System Administrator's Reference Manual

pwr(8)
RM400 only

NAME

`pwr -` set machine-specific system attributes

SYNOPSIS

`/sbin/pwr` [`–a` [*a_flag*]] [`–w` [*w_value*]] [`–s` [*s_value*]] [`–p` [*p_flag*]]

DESCRIPTION

The `pwr` command sets some system attributes that are required with systems that operate a UPS (Uninterruptible Power Supply) on a V24 interface.

The following attributes are supported:

`Autopoweron`
: The system executes an automatic poweron after being powered off as a result of a shutdown caused by a power failure. This system attribute can only be expected if the shutdown has reached the point where the operating system branches to the "init 0" or "init 6" state.

`Wait on Poweroff`
: A wait time is specified to avoid

  a) the system restarting before the UPS has interrupted supply to the system,

  b) the system waiting endlessly for this interrupt if the UPS has not actually interrupted supply to the system

  During this wait time, the system pauses in an inactive state but is not susceptible to a power failure; when this period has expired, the system powers down. If you are using a 9071 type UPS, this wait time should be at least as long as `ups_downdelay` [see `pupsd(8)`].

`Save boot`
: The system can be halted for a certain period of time while it is being booted to allow the batteries to be loaded.

`Powerfail`
: The system is prepared for a shutdown as a result of a power outage.

OPTIONS

`–a` Sets the type of restart.

a_flag is `n`, if a restart is not required.

a_flag is`y`, if a restart is required (thus, for example, `–a y`).

`–w` Sets the period of time during which the system is to wait until it is powered off or powered on again. The unit of time for the value *w_value* is seconds (therefore, for example, `–w 180`, if a value of 3 minutes was set for `ups_downdelay`).

`–s` Sets the period of time during which the system is to wait during a restart (following a power outage), until the batteries have been loaded. The unit of time for the value *s_value* is minutes (therefore, for example, `–s 10` gives the batteries a load time of 10 minutes, before the system finishes the restart).

`–p` Indicates that a powerfail has occurred.

p_flag is `n`, if, for example, a powerfail no longer exists (or this system attribute is not to be used).

p_flag is `y`, if a powerfail is pending and the system behavior described is to be used (thus, for example, `–p y`).

Caution: Only the last occurrence of the respective option is considered (from left to right).

EXAMPLES

```
pwr –a y –w 180 –s 15 –p y
```

Assume there is a Power-Fail and you would like to use Autopoweron. The system is to wait 180 seconds and delay the restart for 15 minutes after power has returned to give the batteries time to recover from the last interrupt phase.

FILES

```
/dev/pwr
```

SEE ALSO

_upstab(4), pupsd(8).

System Administrator's Reference Manual

pyr_iosall(8)

NAME

`pyr_iosall` - device-independent log entry formatter

SYNOPSIS

`/var/adm/error/bin/pyr_iosall` [–v] [*start_no* [*end_no*]]

DESCRIPTION

Attention: Please refer also to `errord(8)` for context.

`pyr_iosall` is the device-independent log entry formatter for preparing error files decompressed with `zcat(1)` , or error messages from SCSI devices recorded in these files. In contrast to the device-specific filters, such as `pyr_iosdisk`, `pyr_iostape` etc. (see **FILES**) a complete list of all error messages is produced, from which, in particular, the sequence of messages with the same time stamp can be recognized.

`pyr_iosall` will supply either an outline view of the error messages for overview purposes, or a detailed view of the error messages in accordance with the device-specific filters.

Specific information can be output quickly, particularly for the detailed view, by specifying a start and end number (the preparation time per 1 MB of compressed data is approx. 20 seconds for the outline view and approx. 5 minutes for the detailed view).

OPTIONS

–v (verbose) Detailed view. The error message is output in accordance with the output of the device-specific filters `pyr_iosdisk`, `pyr_iostape` etc. (see **EXAMPLES**).

A one-line outline view is output by default, comprising:

- serial item number
- date/time of message
- device name
- type of error severity
- `cmd:` = SCSI command (hex)

as well as, if available:

- `sk:` = SENSE KEYs (maximum 4, hex)

or

- `cs:` = Controller status (hex)

start_no end_no

Output entries from/to.

If *end_no* is missing, only the start number *start_no* (i.□e. only 1 entry) is output.

If –1 is given for *end_no*, this means output to end.

DIAGNOSTICS

`0` Command executed successfully

`1` Error

`2` Syntax error

ERROR MESSAGES

`can't create <tempfile>`

The file *tempfile* could not be created.

`open for <device-specific formatter> failed`

A specific filter (`pyr_iosdisk`, `pyr_iostape` etc.) could not be called.

VARIABLES

`PYR_IOS_PATH`

Path of the log entry formatter.

The local `PYR_IOS_PATH` variable can be used to set a current path for the device-specific filter (the default here is `/var/adm/error/bin`). This enables evaluation of the error files from other systems and different releases, provided the associated `/var/adm/error/bin` directory was transmitted along with the error files.

Note:

With releases prior to 5.42, the entries for SENSE KEY or "Controller status" may be distorted in the outline view because of different structures; if in doubt you should check the detailed view.

EXAMPLES

Command: `zcat errf*|pyr_iosall`

Output:

```
 1: 1995 Oct 27 14:40:57  SINIX-Y 5.43 A0017 RM600: viper
 2: 1995 Oct 27 14:53:31 ios0/sdisk006   HARD_ERROR  cmd: 1b  sk: 02 3a 00 00
 3: 1995 Oct 27 14:53:41 ios0/sdisk006   HARD_ERROR  cmd: 1b  sk: 02 04 01 00
 4: 1995 Oct 27 14:53:45 ios0/sdisk006   HARD_ERROR  cmd: 1b  sk: 02 04 01 00
 5: 1995 Oct 31 18:50:51  SINIX-Y 5.43 A0017 RM600: viper
 6: 1995 Oct 31 21:22:20 ios0/sraid050l1 HARD_ERROR  cmd: 2a  sk: 06 15 01 01
 7: 1995 Oct 31 21:22:20 ios0/sraid050l1 SOFT_WARN   cmd: 2a
 8: 1995 Oct 31 21:37:03 ios0/sraid050l1 SOFT_WARN   cmd: 28  sk: 01 00 00 01
 9: 1995 Nov  1 00:01:12 ios0/sraid050l1 HARD_ERROR  cmd: 2a  cs: f306
10: 1995 Nov  1 00:01:12 ios0/sraid050l1 HARD_ERROR  cmd: 2a  cs: f306
11: 1995 Nov  1 00:01:12 ios0/sraid050l1 HARD_ERROR  cmd: 2a  sk: 06 15 01 01
12: 1995 Nov  1 00:01:12 ios0/sraid050l1 SOFT_WARN   cmd: 2a
```

Command: `zcat errf*|pyr_iosall -v 5 7`

Output:

```
   5:
DATE: Tue Oct 31 18:50:51 1995
sysname: SINIX-Y
nodename: viper
release: 5.43
version: A0017
machine: RM600
m_type:
base_rel:
   6:
log date                  : Tue Oct 31 21:22:20 1995
error type                : ET_HARD - Hardware related log entry
severity                  : ES_ERROR - We were unable to complete the request
kernel facility           : EE_DISK
device name               : ios0/sraid050l1
device vendor/product     : SNI     PXRC
FW Rev. / Serial No.      : V26Z /
Driver Message            : ios0/sraid050l1s11: Write Extended 32 <seek> [7703:5
:0] <dkblk> 3944256 <pblk> 3944256
Host Adaptor Work Queue   : #1
Host Adaptor Status       : Good Status, No Host Adaptor Error
SCSI Status               : Check Condition - An error has occurred
IOPB Command Type         : SCSI Pass-Through
SCSI Command              : Write Extended
REQUEST SENSE data        : SENSE = 0x6 ASC = 0x15 ASCQ = 0x1 FRU = 0x1
```

```
SENSE data means          : Mechanical positioning error
Total REQUEST SENSE data  : 00| 70 00 06 00 00 00 00 98 00 00 00 00 15 01 01 00
                            10| 00 01 00 00 01 00 15 01 00 00 01 03 00 00 00 00
                            20| 0a 45 28 03 51 00 00 2a 80 00 00 00 00 00 00 00
                            30| 00 00 00 00 00 00 20 20 20 20 20 20 43 58 35 33
                            40| 38 30 30 31 39 39 56 32 36 5a 00 00 00 00 00 39
                            50| 31 30 32 53 54 31 35 31 35 30 4e 20 20 20 20 20
                            60| 20 20 20 00 00 00
lpb flags                 : < BUSY, QUED, SENT, CHECK CONDITION >
Instance Code             : 0328450a
FRU device locator        : 01 03 00
   7:
log date                  : Tue Oct 31 21:22:20 1995
error type                : ET_SOFT - Software related log entry
severity                  : ES_WARN - We managed to recover from this error
kernel facility           : EE_DISK
device name               : ios0/sraid050l1
device vendor/product     : SNI     PXRC
FW Rev. / Serial No.      : V26Z /
Driver Message            : ios0/sraid050l1s11: Write Extended 32 <seek> [7703:5
:0] <dkblk> 3944256 <pblk> 3944256
Host Adaptor Work Queue   : #1
Host Adaptor Status       : Good Status, No Host Adaptor Error
SCSI Status               : Good Status
IOPB Command Type         : SCSI Pass-Through
SCSI Command              : Write Extended
lpb flags                 : < BUSY, QUED, SENT >
```

FILES

`/var/adm/error/bin/pyr_iosdisk`
: Filter for disks

`/var/adm/error/bin/pyr_iosgen`
: Filter for Sgen drivers

`/var/adm/error/bin/pyr_iosjuke`
: Filter for jukeboxes

`/var/adm/error/bin/pyr_iosraid`
: Filter for RAID devices

`/var/adm/error/bin/pyr_iostape`
: Filter for tapes

`/var/adm/error/bin/pyr_iosadp`
: Filter for ADAPTEC-SCSI controllers (RM400 only)

`/var/adm/error/bin/pyr_iosswx`
: Filter for SWX-SCSI controllers (RM400 only)

SEE ALSO

zcat(1), errord(8).

System Administrator's Reference Manual

reinitdev(8)

NAME

`reinitdev` - reinitialize a SCSI device, bus, or controller

SYNOPSIS

`/etc/opt/havail/reinitdev` [`-s`] *devname*

`/etc/opt/havail/reinitdev` [`-s`] [`-n`] *portname*

`/etc/opt/havail/reinitdev` [`-s`] [`-n`] *ctlrname*

DESCRIPTION

`reinitdev` performs a reinitialization of SCSI software and hardware. If `-s` is given, the work will be done silently and without much prompting.

The `reinitdev` command is therefore necessary if the specified hardware is to be initialized. It is used by the OBSERVE high availability application, whereby SCSI buses can be switched from one system to another system. As a result of this, problems can occur on the switched SCSI bus, which can only be solved by resetting the bus or the controller.

devname is the SCSI device name as it is displayed with `autoconf(8)` . Example: `ios0/sdisk033`.

portname is the SCSI bus or port name as it is displayed with `autoconf(8)` . Example: `ios0/sport05`.

ctlrname is the name of the SCSI controller as it is displayed with `autoconf(8)` . Example: `ios0/scon07`.

In addition, the `reinitdev` command is used to perform a device localization, i.e. the assignment of the SCSI bus to the peripherals cabinet. Special hardware requirements must be fulfilled in this case: For example, the peripherals cabinet must support localization.

Localization is only performed if either *portname* or *ctlrname* is specified. Localization is suppressed with the `-n` option.

OPTIONS

`reinitdev` *devname*

Reinitializes a single device. This command performs a hard reset to the specified device. The utility probes to ensure the specified device still exists and then reinitializes all of the software structures associated with that device.

This command has no effect on any other device, the port, or the controller.

`reinitdev` *portname*

Reinitializes a chain of devices on the same SCSI bus. This command performs a hardware reset to the specified SCSI bus, which causes a hard reset to all the devices attached to this bus. The SCSI bus is probed following the reset. All of the software structures for all devices on the bus are reinitialized. Device localization [see `scan_port(8)`] is performed, if possible, for this bus.

This command has no effect on the controller or any devices attached to neighboring buses.

`reinitdev` *ctlrname*

Reinitializes the SCSI controller card and all attached devices. This command performs a hard reset to the controller which causes a hard reset to all the devices attached to that controller. After the reset, the controller is probed to verify the existence of the controller and its devices. All of the software structures associated with that controller and its devices are reinitialized.

Device localization is performed [see `scan_port(8)`], if possible, for all of the SCSI buses attached to this controller.

NOTES

`reinitdev` cannot be performed on the device, its bus, or its controller containing the root file system or the primary swap area.

Parallel activity by `reinitdev`/`changedev` is only permitted using different SCSI buses.

`reinitdev` cannot be performed on a device that is not displayed with `autoconf(8)`.

The `–n` option should only be specified for SCSI buses, which are known not to support device localization.

SEE ALSO

`devname(7)`, `autoconf(8)`, `changedev(8)`, `scan_port(8)`.

System Administrator's Reference Manual

repair(8)

NAME

repair - provide error information in the case of repairs

SYNOPSIS

repair [–o] –n *component* [–p *physical_ID*] [–l *logical_address*] [–s *location*] –a *author* [–c *customer*] [–A *order_no.*] [–L *LED/LCE_info*] –t "*text_or_name_of_textfile*" [–O *output_device*] [–h]

DESCRIPTION

The repair command prepares information that is already available in the system for the purpose of repairs and then provides this information in compressed form in $DUMPDIR/*name* [see dumpsave(4)].

The repair command invokes system and component-dependent shell scripts which have been created by the system administrator in /etc/default/repair/repair.*component*. The name of the system-dependent script is defined with /etc/default/repair/repair.default, whereas the component script is a composite name (/etc/default/repair/repair.*component*, see –n option). Examples of names of component-dependent scripts include:

```
/etc/default/repair/repair.CPM-168400
/etc/default/repair/repair.MBP-29902
/etc/default/repair/repair.DISK
```

The system administrator can create or edit these shell scripts.

If there is no component-specific shell script, the system-dependent shell script /etc/default/repair/repair.default is invoked.

When this shell script is invoked, the directory name ($1) where the files are to be stored is specified with the repair command. All output from the script must be piped to files ($1/*filename*). Output to stderr/stdout is sent to /dev/null.

These are some of the entries that the shell scripts can contain:

```
# Invoking system commands:
autoconf            > $1/autoconf
showconf -l         > $1/showconf
spbuslog            > $1/spbuslog
# Outputting system tables:
cat /etc/device.tab > $1/device.tab
cat /etc/dktab      > $1/dktab
cp /etc/daps/daps/*.daps $1
cp /etc/daps/daps/*.def $1
zcat /etc/osm*.Z    > $1/osm
```

The options allow you to submit information such as component name, physical Id, logical address, location in system, author, order number, customer name, LED display, and comment. □
Dumps already generated can only be overwritten following interaction (see –o option). □
The return value is the directory name of the generated partial dump.

OPTIONS

–o No interaction if the dump is available; the old version is overwritten.

–n *component*
Name of defective component.

–p *physical_ID*
Serial number of defective component.

–l *logical_adddress*
Logical address of component.

`-s` *location*
: Location in system.

`-a` *author*
: Name of the author of the messages.

`-c` *customer*
: Name of customer.

`-A` *order_no.*
: Order number.

`-L` *LED/LCE_info*
: Status of LED/LCE display.

`-t` "*text_or_name_of_textfile*"
: Short error description or name of text file.

`-O` *output_device*
: Writes output to *output_device* (e.g. `/dev/ios0/rsfdisk001ft`).

`-h`
: Displays syntax of `repair`.

EXAMPLES

Creating the repair information under `$DUMPDIR`:

```
repair -n disk47 -a author -t "Bad SCSI disks"
```

FILES

`/etc/default/dumpsave`
`/etc/default/repair/repair.default`
`/etc/default/repair/repair.`*component*

NOTES

Repair information can also be generated, managed and analyzed from the diagnostic menu.

SEE ALSO

`dumpsave(4)` .

System Administrator's Reference Manual

sash(8)
RM400 only

NAME

`sash` - stand-alone shell used to bootstrap Reliant UNIX on an RM400

SYNOPSIS

The SASH is invoked from the RM400 firmware, manually or automatically.

DESCRIPTION

The Stand Alone Shell (SASH) is used to boot the Reliant UNIX operating system. The SASH is loaded into main memory by the RM400 firmware from a specified hard disk, CD-ROM, disk, or tape. Commands are then processed by the SASH to load the operating system. Control is passed to the operating system when the load operation is completed.

SCSI I/O (IOS) disk, CD-ROM and tape drives can be used as boot devices. Alternatively LAN can be used to boot the SASH. Any disk drive containing a `ufs` file system can then be used by the SASH to boot any kernel executable file on that device. A previous mount of that device is not necessary.

Tape devices are generally used to install the operating system software. It is possible however, to boot only a kernel from a CD-ROM or tape device. This is useful for system recovery if the boot disk has been corrupted, preventing the SASH from reading the disk.

After loading the kernel image, an argument vector is created in memory in the first page (4k) in front of the kernel text. This is where options such as the Reliant UNIX root device, root file system type, etc. are passed to the operating system. Once this vector has been created, the SASH launches the kernel by branching to its entry point. Eventually, the SASH is cleared from memory by the operating system.

At startup, the SASH tries to find a special file, called `STARTUP`, at various locations for diagnostic purposes. When the file is found, normal startup is canceled and a special part of the SASH, the test-and-diagnostics shell (TDSH/TDFSH) is activated. Normally this is done only when a system technician runs diagnostic tests. The file must reside in a special file system and is looked after at floppy and hard-disk (swap-partition). If the file is not found, normal startup continues until the SASH command parser is entered or the Reliant UNIX operating system is successfully loaded.

The SASH command parser issues a `sash:` prompt while the RM400 firmware issues a `>>` prompt. The commands of the SASH are listed below, giving their syntax and an explanation of all of the optional and required arguments.

At almost any time, pressing ^D or ^C will cause the SASH to fall back into the RM400 firmware. It can then be reloaded again from that point.

SASH Commands

The following list describes the commands and options provided by the SASH, as identified by the `sash:` prompt. The commands are listed in alphabetical order. The named environment variables are described in `nvram`(7) .

Any *number* or *address* in the list below may have one of the following formats:

Format	Description
`decbase`	*decbase* is a number consisting of decimal digits (0..9)
`0x`*hexbase*	*hexbase* is a number consisting of hexadecimal digits (0..9a..f)
`0`*octbase*	*octbase* is a number consisting of octal digits (0..7)

Example:

The numbers `0xff`, `0377` and `255` resemble the same number.

Any *range* mentioned in the list below specifies a memory range and may have one of the following formats:

Range	Description
base	Specifies the contents of the memory address *base*.
base#	*count*
base:*limit*	Specifies the contents of the memory addresses starting at *base* and ending at *limit*.

Example:

The ranges `0xa0400000#36` and `0xa0400000:0xa0400020` represent the same memory range beginning at `0xa0400000` (hex) and ending at `0xa0400020` (hex) inclusively.

Any *length* mentioned in the list below specifies a length and may have one of the following characters:

Character	Length
`-b`	Byte (8 bits)
`-h`	Halfword (16 bits)
`-w`	Word (32 bits) (default)

- `auto`

  Performs autoboot, i.e. loads the file specified in the environment variable `bootfile`. The device from which this bootfile is loaded must already be part of `bootfile`.

  Example:

  If the environment variable `bootfile` contains `dkncr(0,0,10)sash`, the SASH will be loaded from hard disk `#0`, partition `#10`.

- `boot` [`-n`] [`-f` *file*] [*args*]

  Loads and executes a file from a specified device.

  `-n` For SNI internal development use only. The executable is loaded but not launched. After the SASH has loaded the executable, it returns to the command parser. A subsequent boot command must be used to load and/or launch another executable. This command is useful when an executable is loaded in addition to a debug monitor. An executable loaded with this option may be launched later with the `go` command.

  `-f` *file*

  Specifies the file name to be booted. If no file name is given, the `boot` command uses the content of the environment variable `bootfile`. The file name itself may contain the name of the device where the file resides. If no device is given, the environment variable `path` is prepended to the file name.

  The file name may have one of the following formats:

Device	Format
Tape	`tpncr(`[*host*_
	or
	`tpncr(`[*host*_
Disk	`dkncr(`[*host*_

  If one of the optional arguments is not given, its value is set to 0.

  args Arguments which are passed to the booted image when immediately launched. Arguments which begin with "–" must be prepended with another "–" to tell the SASH command parser that it is not a boot option; this extra dash is removed before the argument is passed to the program.

Example:

The following example loads the file `unix` from partition#0 of disk#0 which is connected to host_adapter#0 and executes the file with run level S:

```
sash: boot -f dkncr(,,)unix initarg=-s
```

- `cat` [*file1 file2 ... fileN*]

  Displays the contents of the listed file(s).

  If a given file name does not contain a device specification, the environment variable `path` is prepended to the file.

- `cp` [`–b` *blocksize*] [`–c` *count*] *file1 file2*

  Copies the contents of *file1* to *file2*.

  file1 is the file to be copied, and *file2* is the file to which it is to be copied *file1*. *file1* and *file2* may be raw devices.

  While this command is being executed, a period (.) is printed on the screen for each record that is transferred.

  `–b` *blocksize*

  blocksize gives the amount of bytes which are copied at each read/write access from *file1* to *file2*. *blocksize* may be followed by one of the following characters, meaning:

Character	Description
`b, B`	*blocksize* is multiplied with 512
`k, K`	*blocksize* is multiplied with 1024 (1 KByte)

  NOTE: There must be no spaces between *blocksize* and the following multiplying character!
  If `–b` is not specified, then the blocksize defaults to 512 bytes.

  For raw devices the blocksize should be an integral multiple of the device's physical record length.

  `–c` *count*

  Specifies the maximum byte count to be transferred. If `–c` is not specified, the copy terminates at EOF of *file1*.

  Example:

  To copy *file3* (the fourth file, zero is the first file) on the streamer tape to partition 1 of the disk, type:

  ```
  sash: cp -b 16k tpncr(,6,3) dkncr(,,1)
  ```

- `disable` [*console_dev*]

  Does not allow input from and output to the specified console device. If you use `disable` without arguments, then the current set of enabled console devices is displayed.

  Example:

  `disable tty(1)` disables the serial remote port `tty(1)`.

- `dump` [*length*] [*format*] [*range*]

  Formats and displays the contents of memory.

  length

  Is a character that specifies the length of the location to dump.

  format

  Is a character that specifies how to display the contents of memory as follows:

Character	Format
`–B`	binary
`–c`	ASCII character

–d	decimal
–o	octal
–u	unsigned decimal
–x	hexadecimal (default)

range Is a valid address in memory with optional decimal delimiters.

Example:

The following example shows a *base*#*count range* specified in halfwords. The default for the *format* of hexadecimal is used because no argument was specified.

```
sash: dump -h 0xbfc04000#5
0xbfc04000: 8dce 514 6 6900 1a3
```

- `enable` [*console_dev*]

  Allows input from and output to the specified console device from the SASH command parser.

  If you use `enable` without arguments, then the current set of enabled console devices is displayed.

  Example:

  `enable tty(1)` enables the serial remote port `tty(1)`.

- `end`

  Ends the command parser of the SASH and falls back to the caller of the SASH.

  The return value is the last return value from a program booted and executed by the SASH.

- `fill` [*length*] [–v *val*] [*range*]

  Fills the contents of a specified range of addresses with *val*.

  length
  Is a character that specifies the length of the location to fill.

  val Is a number with which to fill memory.

  range Is a valid address in memory with optional decimal delimiters.

  Example:

  The following example fills the contents of memory with a word with the value `0xff00ff00` starting at `0xaffffffff` for 10 32-bit locations:

  ```
  sash: fill -w -v 0xff00ff00 0xaffffffff#10
  ```

- `g` [*length*] [*address*]

  Displays the contents of a single memory location in decimal, hexadecimal, and ASCII character formats.

  length
  Is a character that specifies the length of the location to dump.

  address
  Is either a single memory location or a register name.

  Example:

  ```
  sash: g 0xa0013cfc
  address       value in decimal   value in hex   value in ASCII□
  0xa0013cfc:   -192471552         0x8e200000     '\'
  ```

- `go` [*address*] [–c *args*]

  Launches an executable previously loaded into memory by the `boot` command.

  address
  Specifies the address at which to begin execution of a previously loaded image. Useful for choosing the desired address where several images are loaded. If omitted, the image most recently loaded is

launched.

`-c` *args*
Used for passing arguments to the program. All arguments following this options are passed to the program; there must be no additional dash prepended to arguments beginning with a dash.

- `help` [*commandlist*]
- `?` [*commandlist*]

Displays the syntax for all the commands in *commandlist*.

commandlist
Is one or more commands separated with a space.

If you do not specify a command list, then the `help` command displays the syntax for all commands. You may enter a question mark (?) instead of `help`.

- `init_tod` [*secs*]

Initializes the time-of-day chip.

secs Is the number of seconds since 1972. If *secs* is omitted, it defaults to zero.

It is very important that the time-of-day chip is running; otherwise, the operating system will not work properly. This command is normally executed at the factory.

NOTE: This command works only in bootmode "d".

Example:

The following procedure gives the recommended method of initializing the time-of-day chip.

1. Type the `init_tod` command without any arguments.
2. Run the `date(1)` command after the operating system has been booted.

- `load` [*char_device*]

For internal Siemens use (system technicians) only.

The `load` command allows you to load memory over a serial line connection from a system running the program `tip`.

- `p` [*length*] [*address*] [*value*]

Puts a specified value into a single memory location.

length
Is a character that specifies the length of the location to fill.

address
Is the address of a single memory location.

value Is the value to which you want to set the address.

- `printenv` [*varlist*]

Displays the value of the environment variables in *varlist*.

varlist
Is a specific environment variable or more variables separated by a space. If no environment variables are specified, `printenv` shows all currently defined environment variables.

Example:

The following example prints the value of the environment variable `netaddr`.

```
sash: printenv netaddr
netaddr = 141.29.51.3
```

- `pr_tod`

Prints the contents of the time-of-day register from the time-of-day chip.

The displayed value is the number of seconds since 1972, printed in hexadecimal.

You can determine whether the time-of-day chip is functioning by running this command twice. If you run the command once and then run it again 5 seconds later, the value displayed the second time should be about 5 more than the first time. If the displayed value does not change or if the value decreases, then run the `init_tod` command.

Example:

The following example shows the hexadecimal value for the number of seconds since 1972.

```
sash: pr_tod
tod = 0x24bdda99
```

- `setenv` [*var*] [*value*]

  Creates a new environment variable or changes the value of an existing environment variable.

  var Is the environment variable you want to set.

  value Is the value at which you want to set the variable.

  Environment variables are represented as ASCII strings. The current values of the environment variables are passed to programs booted by the SASH.

  Example:

  The following example sets the environment variable `rbaud` to 9600.

  ```
  sash: setenv rbaud 9600
  ```

- `sload` [–a] [–b] [*console_dev*]

  Accepts a subset of the Motorola S-record protocol. The record types accepted are 0, 3 and 7.

  –a Disables the sending of acknowledgements. If you do not specify `–a`, `sload` replies with an ASCII ACK to each S-record received that has a valid checksum. `sload` replies with an ASCII NACK for records that have incorrect checksums.

  –b Directs `sload` to expect hybrid ASCII-binary records. If you specify `–b`, `sload` expects hybrid ASCII-binary records, which are produced by the `sdownload` program. Using these hybrid ASCII-binary records roughly doubles the throughput of serial transfers.

  console_dev
  The name of the console device.

  Use the SPP command `convert` to produce S-record images.

  Example:

  The following example shows the command to download S-records via `tty(1)`.

  ```
  sash: sload tty(1)
  ```

  `sload` is now waiting for the S-records sent by `sdownload` over the `tty(1)` port.

- `spin` [[–i *count1*] [–c *count2*] [–v *value*] –{r|w} –{b|h|w} *address*]

  Performs load or store operations on the specified address using the specified byte, halfword or word width.

  –i *count1*
  count1 is an iteration counter that executes each "subgroup" as specified, then repeats it *count1* times.

  –c *count2*
  count2 is a numeric value that specifies the repetition or copies that apply to all subsequent reads.

  –v *value*
  value indicates a numeric value for writes that applies to all subsequent writes.

  address
  Is the hexadecimal address you want to look at.

  count defaults to 1 and *value* defaults to 0. A negative count is interpreted as infinity.

  You can specify a sequence of reads and/or writes of byte (`b`), halfword (`h`) or word (`w`) width with

combinations of –r and –w options.

Example:

The example below writes the value 9200 to the location 0xa0300000 for 100 iterations, reads location 0xa0300004 100 times, and then repeats the entire operation 20 times.

```
sash: spin -i 20 -c 100 -v 9200 0xa0300000 -r 0xa0300004
```

- `tdsh`

  For internal Siemens use (system technicians) only.

  A special test-and-diagnostic shell is entered. This point describes how to return to the `sash` prompt.

If you see	enter
`&`	`e` followed by RETURN
`%`	`!;e` followed by RETURN

- `unsetenv` [*var*]

  Deletes an existing environment variable.

  var Is the specified environment variable.

  Example:

  The following example unsets the `rbaud` variable.

  ```
  sash: unsetenv rbaud
  ```

- `warm`

  Performs a warm boot of the system. Use this command to see a core dump of memory before rebooting.

  The SASH monitor looks for the restart block at location 0x400 in physical memory. It considers the restart block valid under these conditions:

  - (offset 0) word 0 = 0xfeedface
  - (offset 8) word 2 = 0
  - (offset 12) word 3 = the 32-bit two's complement sum of the first 32 words to which word 1 (offset 4) of the restart block points.

  The `warm` command uses the default values contained in the PROM's restart block (shown below):

```
/*
 * Restart block -- monitor support for "warm" starts
 *
 * prom will perform "warm start" if restart_blk is properly set up:
 *      rb_magic == RESTART_MAGIC
 *      rb_occurred == 0
 *      rb_checksum == 2's complement, 32-bit sum of first 32, 32-bit words
 */
 define RESTART_MAGIC   0xfeedface
 define RESTART_CSUMCNT 32         /* chksum 32 words of restart routine */
 define RESTART_ADDR    0xa0000400 /* prom looks for restart block here */
 define RB_BPADDR       (RESTART_ADDR+24)  /* address of rb_bpaddr */

 ifdef
LANGUAGE_C
 struct restart_blk
 {
     int  rb_magic;            /* magic pattern */
     int  (*rb_restart)();     /* restart routine */
     int  rb_occurred;         /* to avoid loops on restart failure */
     int  rb_checksum;         /* checksum of 1st 32 wrds of restrt */
```

```
        char *rb_fbss;            /* start of prom bss and stack area */
        char *rb_ebss;            /* end of prom bss and stack area */
        /*
         * These entries are for communication between the debug monitor
         * and the client process being
debugged
         * NOTE: a return value of -1 from (*rb_vtop)() is distinguished
         * to indicate that a translation could not be made.
         */
        int  (*rb_bpaddr)();      /* breakpoint handler */
        int  (*rb_vtop)();        /* virtual to physical conversion rtn */
        /*
         * config table goes here
         */
 };
 endif LANGUAGE_C
```

SEE ALSO

nvram(7), prom(8).

System Administrator's Reference Manual

scan_port(8)

NAME

`scan_port` - determine assignment of SCSI bus to peripheralscabinet

SYNOPSIS

`scan_port` [`-f`] *portname*

`scan_port` [`-f`] *ctlrname*

`scan_port` [`-f`] [`-p`] `-a`

DESCRIPTION

The `scan_port` command is used to perform a device localization, i.e. the assignment of the SCSI bus to the peripherals cabinet is established. The defined assignment is recorded in `HW_CONFIG`. This information is needed by CONFIG for (graphical) system display purposes.

Device localization is performed at system start for all SCSI buses (ports). In addition, the `reinitdev(8)` command performs localization for every reinitialized bus.

The `scan_port` command issues SCSI bus resets to the ports during the localization and then uses EIP queries to ascertain assignments for all peripherals cabinets. Since the time required for localization almost quadruples with the size of the system configuration, only unscanned and reconfigured buses are scanned. Data from the scanned SCSI buses is stored in `/etc/scsidevconf.lar`.

OPTIONS

portname
: Device localization is performed for the specified port [see `autoconf(8)`] (e.g. `ios1/sport12`).

ctlrname
: Device localization is performed for all ports of the specified controller [see `autoconf(8)`] (e.g. `ios1/scon12`).

`-a`
: Device localization is performed for all modified ports of all controllers (system start).

`-f`
: Device localization is performed regardless (enforced).

`-p`
: Error messages are simply logged during device localization for all ports. These error messages can also be output by specifying the `-p` option.

NOTES

Device localization is not supported by all peripherals cabinets (e.□g. not by PXRC). This means that the affected SCSI buses cannot be localized and an error message is issued. Localization is also not possible if the EIP cabling/configuration is not correct.

If basic system settings are changed (e.□g. the EIP-PID is changed), `scan_port` must be executed with the additional `-f` (`scan_port -af`) parameter.

WARNING

The `scan_port` command issues SCSI bus resets to the SCSI buses (specified port) during device localization and should therefore not be executed during normal operation.

FILES

`/etc/scsidefconf.lar`

SEE ALSO

`ups(1M)` [only RM600], `autoconf(8)` , `eip2cmd(8)` [only RM400], `reinitdev(8)` .

System Administrator's Reference Manual

scsiutil(8)

NAME

`scsiutil` - SCSI utility for model 3444/3448 tape drives

SYNOPSIS

`scsiutil [-vTEF] [-Sis] [-ed] [-mju] [-l` *slot* `] -f` *devicefile*

`scsiutil [-v] -E`

DESCRIPTION

The purpose of `scsiutil` is to facilitate special features of Pyramid's half inch tape cartridge drives, models 3444 and 3448.

OPTIONS

The primary options are: [`isedmjul`]. At least one of the primary functions must be used to avoid an error message. However, you may use only the `-v` or the `-E` option to get verbose usage messages or examples, respectively. The options provide functionality as described below:

`-v` Provides verbose messages.

`-T` (Trace) Shows the sequential actions of `scsiutil`. This option may be too verbose for normal use.

`-E` (Example) Shows examples of how to use `scsiutil`. This option should be used alone, or with the `-v` option, to get both the verbose usage message and the examples.

`-F` (Force) This option allows the user to override the processing which prevents use of an unrecognized device. This option is not recommended for normal use. If used on an unrecognized device, it may cause an I/O error message to be logged at the system console.

`-S` This option allows the user to get the SCSI `Mode Sense` data for the given device. This option is designed for SCSI device trouble analysis, and is probably of no use to most users, unless they are fluent in SCSI.

`-i` Show the device's SCSI `Inquiry` response data, consisting of SCSI Vendor ID, Product ID, and Firmware Revision. The output is of the form *Name*=*Value*. This form allows the use of `eval(1)`, to evaluate the assignment strings into your shell environment. However, if you do use `eval(1)`, you must not use the `-v` option, because the output is shifted to the right for aesthetic purposes.

The following is a sample output of *Names* and *Values* that are set with this option:

```
VendorID='FUJITSU '
ProductID='M1016B  M2481B  '
FirmwareRev='0014'
```

`-s` Get the tape drive and autoloader statistics. This provides information on whether there is a tape cartridge or tape magazine loaded, the autoloader's loading mode, the size of the magazine, the position of the magazine (if installed), and whether the magazine can be ejected automatically. The output is of the form *Name*=*Value*. This allows use of `eval(1)` to pull the variable assignment into your shell script (see below).

The following is a sample output of *Names* and *Values* that are set:

```
TapeLoaded=False
AutoLoaderMode=AutoLoad   (always in AutoLoad mode)MagazineEject=Enabled
 (always enabled)MagazinePresent=True      (not necessarily loaded)MagazineS
ize=10           (tape capacity)MagazinePosition=0        (magazine is eject
ed)SlotsWithTapes=1:2:3:4:5:6:7:8:9:10
```

`-e` Enable compression, if the device supports compression functionality. If a tape is loaded, it must be at BOT (Beginning Of Tape).

`-d` Disable compression, if the device supports compression functionality. If a tape is loaded, it must be at BOT.

`-m` Load the tape magazine. After loading the magazine, it will then load the first available tape.

`-j` Eject the tape magazine. If a tape is loaded, it will unload the tape before ejecting the magazine.

`-u` Unload the tape from the drive, and load the next tape. If the next tape does not load automatically, verify that the autoloader on the front of the tape drive is in `AUTO` mode. After unloading the last tape, the magazine will also be ejected.

`-l` *slot*
Load the tape from the specified *slot* of the tape magazine. Tape slots are numbered sequentially from top to bottom, starting with slot 1.

`-f` *devicefile*
Specify the character tape device file. This is a required parameter.

EXAMPLES

To see examples, run `scsiutil`, with the `-E` option, as shown below:

```
scsiutil -E
```

NOTES

There are few restrictions on the combinations of options that may be specified in a single invocation, however, they will be performed in a pre-determined order. For example, if both load and eject magazine options are selected, it will load the magazine before it ejects it.

In some cases, preparatory operations will be performed, if the device is not ready for a given request. For example, if you request to load a specific tape when the magazine is not loaded, then `scsiutil` will first attempt to load the magazine.

In cases where the requested operation is already complete before the request is made, then `scsiutil` will not repeat the same operation. For example, if tape 2 is loaded, and a request is made to load tape 2, then `scsiutil` will not attempt the operation.

For the compression options [`e`,`d`], some restrictions apply. Most model 3444 or 3448 tape drive units do not support compression. If compression is not supported, an attempt to enable compression will fail (with an error message). This is not an error. If you did not explicitly order the compression option, then your unit will not support compression.

The options to enable or disable compression should only be used when the tape is at BOT, or there is no tape loaded.

Depending on your device's compression firmware revision, you may see a message from `scsiutil` that it is retrying to enable compression. This is normal, it is simply attempting to perform the same function again, but in a different way.

The output of the `-s` and `-i` options are of the form: *Name*=*Value*. This form allows use of `eval(1)` to duplicate the variables in your shell environment. Once in your shell environment, you can reference these variables from a shell script if desired. Note that the `-v` option is incompatible with the `eval` function of the shell, so do not use the `-v` option with `eval`.

If there is no tape in the tape drive unit when you run `scsiutil`, there will be a message sent to the system console including the phrase: "Unit Not Ready". This is normal, because `scsiutil` performs its operations whether or not the device is technically "Ready" to be accessed.

When you use `-s` to Sense Cartridge data, if the value of the `MagazinePosition` variable is "0", then the magazine is not loaded. It may be sitting in the top of the Automatic Cartridge Loader (ACL). In this case, the variable `SlotsWithTapes` will be empty.

This utility is only designed to perform correctly on the models 3444 and 3448, and not supported on any other tape devices. However, if it is used on other SCSI tape devices to get SCSI `Inquiry` data, access may fail if there is no media in the tape drive being accessed.

If the requested operation terminates with no tape in the drive, then the standard operating system tape `close(2)` function will result in an attempt to execute certain tape operations which must fail, because there is no tape in the drive. The result is that a tape "Unit Not Ready" message will be sent to the system console. This may be somewhat annoying, but is expected operation under these conditions.

SEE ALSO

`eval(1)`, `mt(1)`.

System Administrator's Reference Manual

scstat(8)

NAME

`scstat -` system call statistics monitor

SYNOPSIS

`/usr/sbin/scstat –[ z ]a`

`/usr/sbin/scstat –[ z ]p` *pid*

`/usr/sbin/scstat –[ z ]d`

`/usr/sbin/scstat –z`

`/usr/sbin/scstat [ –cste ] [ –o` *output* `] [` *interval* `[`*count* `]]`

`/usr/sbin/scstat [ –u` *user* `] [ –g` *group* `] –x` *command*`]`

DESCRIPTION

The kernel can keep track of which system calls are being executed on each processor and the amount of time that each processor is devoting to the execution of each system call. Unless this capability is activated, the kernel does not keep track of such system call statistics.

`scstat` provides the user the ability to do the following: activate and deactivate the kernel system call monitoring, clear the system call statistics that have been gathered by the kernel, and report the system call statistics that the kernel has gathered.

OPTIONS

The following options may be supplied to `scstat` to control the state of kernel system call monitoring:

`–a` Activates the kernel system call monitoring for all processes.

`–p` *pid*
: Activates the kernel system call monitoring only for the process that has a process ID of *pid*.

`–d` Deactivates the kernel system call monitoring.

`–z` Clears the system call monitoring statistics that the kernel has gathered.

`-x` *command*
: Executes *command* with system call monitoring activated.

`–u` *user*
: Sets the user ID before executing *command*.

`–g` *group*
: Sets the group ID before executing *command*.

Note that the `–a`, `–p`, and `–d` options are mutually exclusive, and that these three options are only available for use by the superuser.

When either no options or the following options are supplied, `scstat` samples the system call statistics being gathered by the kernel every *interval* seconds and dynamically updates a report on the terminal screen *count* times. If the output is to be redirected to a file, you must use the `–o` option. The default *interval* is 5 seconds and the default *count* is 10000. Note that this can only be done while kernel system call monitoring is active.

`–c` Reports detailed system call activity on a per-CPU basis. The default is to report detailed system call activity collectively for all processors.

`–s` Reports cumulative system call activity since kernel system call monitoring has been activated. This option causes the total number of system calls, as opposed to the system call rate, to be reported.

`–t` Sorts the detailed system call information being reported by the percentage of time for each system call instead of by total number of system calls or system call rate.

`–e` Includes statistics for `exit(2)`.

–o *output*

Outputs the system call report information to the specified *output* file instead of the terminal screen.

The first line of the report is a summary of the system call activity that occurred during the time *interval*.

Here are two examples:

If activated with –p 266 and the –s option has been specified, and the machine has two processors, the first line of the report may appear as follows:

```
(pid 266) Total calls 75. By CPU:
(0) 25   (1) 50
```

This summary shows that process ID 266 has executed a total of 75 system calls, 25 executed on processor number 0 and 50 executed on processor number 1.

If activated with –a and the –s option has not been specified, and the machine has four processors, the first line of the report may appear as follows:

```
Total calls/sec 100. By CPU:
(0) 26  (1) 24  (2) 23  (3) 27
```

This summary shows that all processors collectively are executing 100 system calls per second; 26 calls per second are being executed on processor number 0, 24 calls per second are being executed on processor number 1, 23 calls per second are being executed on processor number 2, and 27 calls per second are being executed on processor number 3.

The remainder of the report provides detailed information about which system calls are being executed, the execution rate or total number of executions, and the percentage of system call time that is being spent for execution. When the –c option is specified, this detailed information is reported for each processor on a separate section of the terminal screen or in a separate portion of the *output* file when the –o option is specified.

EXAMPLES

The following command will activate kernel system call monitoring for all processes:

```
/usr/sbin/scstat –a
```

The following command deactivates kernel system call monitoring and clears the statistics that the kernel has been gathering:

```
/usr/sbin/scstat –dz
```

The following command samples the kernel system call statistics every 10 seconds, for 20 iterations. Each iteration updates the terminal screen with system call rate information for each processor installed in the system, sorted by system call rate:

```
/usr/sbin/scstat –c 10 20
```

The following command samples the kernel system call statistics every five seconds, for 10000 iterations. Each iteration updates the terminal screen with cumulative system call information for all processors, sorted by percentage of system call time:

```
/usr/sbin/scstat –st
```

Note that in the above example, an interrupt will cause program termination so that the full 10000 iterations do not have to be performed.

The following command samples the kernel system call statistics every 20 seconds, for 50 iterations. Each iteration outputs system call rate information to file *output* for each processor installed in the system, sorted by system call rate:

```
/usr/sbin/scstat –c –o output 20 50
```

SEE ALSO

truss(1).

System Administrator's Reference Manual

setinfo(8)

NAME

`setinfo -` set or display device configuration info

SYNOPSIS

`setinfo –l` *pathname*

`setinfo –t` *info* [`–wv`] *pathname*

`setinfo –p` {*info* | `all`}

`setinfo –C` {`l` | `e` | `d`} *pathname*

`setinfo –Cc` {*pathname* | `all`}

`setinfo –T` {`e` | `d`} *pathname*

`setinfo –m` *pathname*

DESCRIPTION

`/sbin/setinfo` describes a device to the kernel. *info* files are composed in a simple language describing device configuration parameters. The language consists of *identifier*=*value* pairs. The identifiers recognized are device specific. A particular identifier expects a value of a particular type.

Several types of values are supported. Values enclosed in single or double quotes are treated as strings. Simple integer values or hex numbers are recognized. Arrays of integers are recognized as a comma separated list of integers enclosed in curly brackets.

For example:

```
dkt_bsize=2048
        # A simple integer
dkt_name='NEC2363'
        # A string value
xmi_addrs={0x1a0,0x1b0,0x12c8}
        # An array of integers
```

The *info* language is compatible with the Bourne shell variable assignment syntax. The output of `setinfo –l` may be used in shell scripts to initialize the value of shell variables.

pathname is the name of a raw device such as `/dev/ios0/rsdisk000s0` or `/dev/ios0/rstape000`.

Some devices, such as magnetic disks, are capable of saving the configuration info so that it will not be lost during a system reboot. These devices only need a setinfo to be done once during installation of the device.

OPTIONS

`–l` Displays the configuration info for *pathname*.

`–t` *info*
: Sets the configuration of *pathname* to *info*. *info* is the name of a file containing *identifier*=*value* pairs describing various descriptive and configuration parameters for the device. If *info* contains a "/", `setinfo` tries to open *info*, otherwise `setinfo` searches for *info* in the directories `/etc/default/diskinfo/setinfo` and `/etc/default/tapeinfo/setinfo`.

`–v` Verbose. Additional display of the configuration data.

`–w` Does not attempt to save the info on the device.

`–p` {*info* | `all`}
: Loads one or all *info* files (see `–t` option) in the kernel.

`–C`{`l` | `e` | `d`}
: By default, the write cache of SNI SCSI disks is disabled for better reliability. This function makes it

possible to `l`ist the state of the write cache or `e`nable or `d`isable the write cache of desired SCSI disks.

Caution: Enabling or disabling of SCSI write caches is only possible on inactive SCSI disks that have not been mounted, to prevent data corruption. This command should be executed typically prior to `fsck` or `mount` start scripts.

`-Cc` {*pathname* | `all`}
Checks if the status of the write cache is compatible with the status in the system file `/etc/default/wcache`. If there are differences, appropriate error messages are output. This procedure can be used to check either individual disks or all disks known in the system.

`-T` {`e` | `d`}
Activates (`e`) or deactivates (`d`) "Tagged Queuing" for SCSI-2 disks. ("Tagged Queuing" means that a disk can accept a number of tasks while other tasks are already being processed.)

Warning: A disk must not be in "used" mode when (`-Td`) is deactivated.

`-m` *pathname*
Initalizes the specified disk with the default attributes. The error handling and cache parameters (mode select parameters) are set here to ensure optimal operation of the disk.

DIAGNOSTICS

Exit status is normally `0`. If an error occurs, exit status is `1`.

NOTE

Some hardware is self describing and therefore does not require a `setinfo` before use.

FILES

`/etc/default/diskinfo/setinfo`
A directory containing disk configuration files.

`/etc/default/tapeinfo/setinfo`
A directory containing tape configuration files.

SEE ALSO

`autoconf`(8), `dkpart`(8), `dksetup`(8), `dktune`(8).

System Administrator's Reference Manual

setup_dbg_kernel(8)
RM600 only

NAME

`setup_dbg_kernel -` activate linked debug kernel for next reboot

SYNOPSIS

`/.utsdbg/etc/conf/bin/setup_dbg_kernel –a`

`/.utsdbg/etc/conf/bin/setup_dbg_kernel –h`

DESCRIPTION

This command copies the linked `MD_unix` debug kernel from `/.utsdbg/etc/conf/cf.d` to `/etc/conf/cf.d`. The name is changed to `MP_unix` in the process and a hard link is set up to `unix`. In addition, the `/etc/.new_unix` file is created which indicates at the next reboot that `/etc/conf/cf.d/unix` is to be activated under `/unix` and `/stand/unix`.

Please refer to the `utsdbg(5)` manual page for an overview of the installation procedure, generation, as well as the activation and deletion of the debug kernel.

OPTIONS

`–a` If the `–a` (activate) option is specified, the functionality is implemented as described above.

`–h` An explanatory syntax message is displayed.

DIAGNOSTICS

Since `setup_dbg_kernel` invokes other commands in order to activate the debug kernel for the next reboot, errors detected with this command are also reported. In general terms, the return code 1 or 10 indicates that `setup_dbg_kernel` has detected an error.

The return code 10 indicates a syntax error when calling `setup_dbg_kernel`.

The return code 1 is used for other errors that have occurred.

All error messages should be comprehensible without further explanation.

NOTES

If a successful `idbuild -M` is executed before the next reboot after the `setup_dbg_kernel` command has been called correctly, the debug kernel brought to `/etc/conf/cf.d` is overwritten by the newly linked kernel. As a result, the debug kernel is not activated at the next reboot. Instead, the newly linked kernel is active.

If you have called `setup_dbg_kernel -a` successfully, but do not wish to activate the debug kernel at the next reboot, you should delete the `/etc/.new_unix` file.

SEE ALSO

`idbuild(1M)`, `utsdbg(5)`.

System Administrator's Reference Manual

sgenconf(8)

NAME

`sgenconf -` configure a generic device

SYNOPSIS

`sgenconf –l`

`sgenconf –d` *index*

`sgenconf –a [ –i` *inqname* `] [ –n` *newname* `]` *devname*

DESCRIPTION

ASCSI device with a SCSI device type which corresponds to one of the standard SCSI device drivers must be entered in the device name table in `/etc/conf/pack.d/sgen/space.c`, if it is to be configured for the generic SCSI driver. For further information on the structure of the device name table, see `sgen(7)`.

It is recommended that you use `sgenconf` when making entries in the device name table.

OPTIONS

`-l` Displays the device name table.

`-a` Adds an entry to the device name table. *devname* is the device node name for the device. For further information on the structure of the device node name, see `sgen(7)`. The device node name should be specified in the form that will be displayed later by `autoconf(8)` (e.g. `ios0/sgen006`).

The manufacturer name and product name can also be specified from the SCSI inquiry data as the *inqname*. In this case, the device is configured for the generic SCSI driver, if both the device node name and the manufacturer name and product name for the device correspond. The manufacturer name and device name must correspond exactly to bytes 8 to 24 from the SCSI inquiry data. The number of blanks between the manufacturer name and the product name is also significant.

In *devname* and *inqname*, `?` represents any character. Both character strings can be shortened. Only the number of characters in the character string are compared.

newname is a freely selectable identification for the device. This identification is displayed by `autoconf(8)` and other administrative commands. If you do not specify *newname*, the character string `Generic SCSI Device` is used as identification.

`-d` Deletes an entry from the device table. *index* is the number of the entry as displayed by `sgenconf -l`.

EXAMPLES

The device with the SCSI ID 6 on the SCSI bus 0 should be configured for the generic SCSI driver:

```
sgenconf -a ios0/sgen006
```

All devices with the manufacturer and product name `ATGCYG 1803`, `ATGCYG 1802`, or `ATGCYG 1602` should be configured for the SCSI driver, regardless of the SCSI bus to which they are connected, and with which SCSI ID:

```
sgenconf -a -i"ATGCYG 1?0" ""
```

NOTES

When configuring generic devices, please ensure that the system disk is not inadvertently configured as a generic device (e.g. `sgenconf -a "ios0/sdisk???"`). Otherwise, the system will not power up the next time it is booted.

SEE ALSO

`sgen(7)`, `sgenio(7)`, `autoconf(8)`, `sgensetup(8)`.

System Administrator's Reference Manual

sgensetup(8)

NAME

sgensetup, sroutsetup - set up special files

SYNOPSIS

sgensetup –U *autoconf-name*

sgensetup | sroutsetup –a *autoconf-name* [–n *symbolic-name*]

sgensetup | sroutsetup –d *autoconf-name* | *symbolic-name*

sgensetup | sroutsetup –l

DESCRIPTION

sgensetup is used to set up a special file for a generic SCSI device in /dev/ios0. If the device has a serial number, a special file with a freely definable name can be created in /dev/sym in addition to the name in /dev/ios0. These so-called symbolic names are particularly useful for devices in cluster configurations [see clos(7)], as well as for devices connected to an FC/SCSI router. The symbolic names are managed globally for the cluster in cluster configurations.

sroutsetup can be used to set up a symbolic name for an FC/SCSI router [see srout(7)].

autoconf-name is the name of the device as displayed by autoconf(8) .

symbolic-name is a freely definable name for the device; *symbolic-name* can also be a relative path name.

OPTIONS

–U A special file is created in /dev/ios0.

–a A special file with a symbolic name is set up in /dev/sym for the device specified by *autoconf-name*. sgensetup assigns the name /dev/sym/gen*x* by default, and sroutsetup the name /dev/sym/rout*x*, where *x* is the smallest decimal number that has not yet been used previously for another symbolic name. This name can be overridden with the additional –n option.

–n Overwrites the symbolic name assigned automatically with –a *autoconf-name*. Only one symbolic name is permitted for each device. If a symbolic name already exists for the selected device, it will be replaced by the new name.

–d The symbolic name for the device specified with *autoconf-name* or *symbolic-name* is deleted.

–l Lists all symbolic names currently assigned in the cluster.

SEE ALSO

clos(7), sgen(7), stape(7), autoconf(8), setinfo(8).

System Administrator's Reference Manual

show_fc(8)

NAME

`show_fc -` show current fibre channel (FC) configuration

SYNOPSIS

`show_fc` [`–n`|`-t`] [`–c` *ctrl-no*]

DESCRIPTION

`show_fc` provides information about all installed FC controllers and connected FC devices.

OPTIONS

If no option is specified, information is output about all FC controllers and devices in a tree structure. A line in the following format is output for every FC controller and every FC device:

`Type`, `No.`, `D_ID`, `L_ID`, `PortName`, `NodeName`, `ULP`, `STAT`

where:

`Type` is the type of device. The following device types are possible:

- `HBA` Host bus adapter
- `FAB` FC fabric/switch
- `DEV` FC device

`No.` is the logical controller number of host bus adapter.

`D_ID` is the FC address (destination ID). The address is specified in hexadecimal notation.

`L_ID` is the FC loop ID. The loop ID is a decimal value. If no loop ID can be specified for a device, "`---`" is output.

`PortName`
is a unique port name worldwide. The output is hexadecimal, whereby the individual bytes in the name are separated by a ":".

`NodeName`
is a unique node name worldwide. The output is hexadecimal, whereby the individual bytes in the name are separated by a ":".

`ULP` is the upper layer protocol. Protocols that are supported by the device. The following protocols are possible:

- `FCP` FC protocol for SCSI
- `---` Not known

`STAT` is the device status. The following states are possible:

- `OK` No error
- `ERR` Device not operational

`–n` If the `–n` option is specified, the output of the title line of the table is suppressed.

`–t` If the `–t` option is specified, the information is output in a tree structure.

In contrast to output in table format, the port and node name are preceded by the `WWN` entry. The port and node name are separated from one another in this case by a ";".

`–c` *ctrl-no*
If the `–c` option is specified, only information about the specified FC controller and about the FC devices connected via this controller is output.

EXAMPLES

Output in table form:

```
show_fc
Type|No.|D_ID     |L_ID|PortName                |NodeName                |ULP|STAT|
 HBA|   0|01.10.00| ---|10:00:00:00:c9:20:26:aa|10:00:00:00:c9:20:26:aa|FCP|  OK|
 FAB|   0|ff.ff.fe| ---|20:00:00:60:69:10:02:0a|10:00:00:60:69:10:02:0a|---|  OK|
 DEV|   0|01.1e.c7|  21|22:00:00:20:37:00:5e:1f|20:00:00:20:37:00:5e:1f|FCP|  OK|
 DEV|   0|01.1e.c9|  20|22:00:00:20:37:00:65:67|20:00:00:20:37:00:65:67|FCP|  OK|
 HBA|   1|01.18.00| ---|10:00:00:00:c9:20:06:31|10:00:00:00:c9:20:06:31|FCP|  OK|
 FAB|   1|ff.ff.fe| ---|20:08:00:60:69:10:02:0a|10:00:00:60:69:10:02:0a|---|  OK|
```

Output as tree structure:

```
show_fc -t
HBA 0, D_ID 01.10.00, L_ID ---, WWN 10:00:00:00:c9:20:26:aa;10:00:00:00:c9:20:26:
aa, ULP FCP, STAT  OK
  FAB, D_ID ff.ff.fe, L_ID ---, WWN 20:00:00:60:69:10:02:0a;10:00:00:60:69:10:02:
0a, ULP ---, STAT  OK
    DEV, D_ID 01.1e.c7, L_ID  21, WWN 22:00:00:20:37:00:5e:1f;20:00:00:20:37:00:5
e:1f, ULP FCP, STAT  OK
    DEV, D_ID 01.1e.c9, L_ID  20, WWN 22:00:00:20:37:00:65:67;20:00:00:20:37:00:6
5:67, ULP FCP, STAT  OK
HBA 1, D_ID 01.18.00, L_ID ---, WWN 10:00:00:00:c9:20:06:31;10:00:00:00:c9:20:06:
31, ULP FCP, STAT  OK
  FAB, D_ID ff.ff.fe, L_ID ---, WWN 20:08:00:60:69:10:02:0a;10:00:00:60:69:10:02:
0a, ULP ---, STAT  OK
```

DIAGNOSTICS

`show_fc` terminates with a return value other than 0 in the event of errors. Error messages are self-explanatory.

SEE ALSO

`autoconf`(8), `showconf`(8).

System Administrator's Reference Manual

showconf(8)
RM600 only

NAME

`showconf` - display the system configuration in terms of the controllers on an RM600

SYNOPSIS

`/sbin/showconf` [`–v`] [`–b` *bus*] [`–n`] [`–c`] [`–t`] [`–m`] [`–h`] [`–M`]

DESCRIPTION

`showconf` returns information on the controllers installed in an RM600 system.

OPTIONS

If no option is specified, information is provided on the controllers of all bus types. The multibus processor MBP of the system cabinet is therefore displayed both for the SP and the MBII bus controllers.

Option –v

If the `–v` option is specified, additional information is displayed in accordance with the other options specified:

`–v` If no other options are specified or with `–b`, the subcomponents of PCI boards are displayed.

`–vh` All boards connected to the respective bus are also displayed with their status.

`–vM` The output format for CP4 boards will be modified such that separate lines with preceding identifiers will be created for each of the `mem_module_state` and `mem_module_size` columns.

Option –b

If the `–b` option is specified, information is only displayed about the controllers located on the corresponding bus (SP, MBII, PCI, X86).

Option –n

If the `–n` option is specified, only the information available in NVRAM when the system was last started, is displayed. The NVRAM is written using the `ldcontroller(8)` command each time the system is brought down, in accordance with the information in the `/etc/controller` file.

If `showconf` is called with no option or with the `–v`, `–b` or `–n` option, a line with the following format is generated for each controller to be displayed. In the case of the `–n` option, the `ag` column is replaced by the `option` column, which contains the controller-specific option values predefined using `/etc/controller` [see `controller(4)`].

The individual fields are separated by ":" and can be empty.

```
ctlr_name : chtyp: ctype : log : serialnumber : bus : cab : busid : \ □
slot : ag : ch : hw : fw : mbp
```

`ctrl_name`
Product name of the controller.

`chtyp`
Channel type, indicates the type of a subcomponent on the board (may be empty) in accordance with the `ctrl_name`.

`ctype`
Controller type (e.g. `lce`, `scsi`).

`log` Logical number of the controller as a decimal number (unique within a controller type).

`serialnumber`
Serial number of the controller as a hexadecimal number.

`bus` Type of the relevant bus (SP, MBII, PCI, X86).

`cab` Number of the cabinet containing the controller.

`busid`
Number of the bus containing the controller.

`slot` Number of the slot on the bus.

`ag` Agent number of the controller on the bus.

`ch` Channel number of the controller (can be empty).

`hw` Hardware revision of the controller as a hexadecimal number.

`fw` Firmware revision of the controller as a hexadecimal number.

`mbp` Logical number of the MBP to be used by this controller to receive jobs (must be on the same MBII).

Option –c

If the `–c` option is specified, information about the cabinets is displayed. A line is generated for each cabinet and the individual fields are separated by the ":" character. A line is structured as follows:

```
cab_type : cab_id : hw_rev : fw_rev : serial : cab_name
```

`cab_type`
Name of the cabinet type (`SYSTEM`, `MBII`, ... `UNKNOWN`).

`cab_id`
Cabinet number.

`hw_rev`
Hardware revision of the EIP as a hexadecimal number.

`fw_rev`
Firmware revision of the EIP as a hexadecimal number.

`serial`
Serial number of the EIP as a hexadecimal number.

`cab_name`
Cabinet name (`SY54`, `BG70`, ...) according to nameplate.

Option –t

If the `–t` option is specified, information is displayed on the status of the Built-In-Self-Tests of the controllers. A line is generated for each controller and the individual fields are separated by the ":" character. A line is structured as follows:

```
ctrl_name : chtyp : bustyp : cab : busid : slot : chan : tstid : tstlev \ □
: tst : tst_stat
```

`ctrl_name`
Product name of the controller.

`chtyp`
Channel type, indicates the type of a subcomponent on the board (may be empty) in accordance with the `ctrl_name`.

`bustyp`
Type of the relevant bus (SP, MBII, PCI, X86).

`cab` Number of the cabinet containing the controller.

`busid`
Number of the bus containing the controller.

`slot` Number of the slot on the bus.

`chan` Channel number of the controller.

`tstid`
Number of the last test performed as a hexadecimal number.

`tstlev`

Level of the last test as a hexadecimal number.

`tst` Status of the last test as a hexadecimal number.

`tst_stat`
Status of the last test in plain language in the form of a list of the following keywords separated by commas: `PASSED`, `FAILED`, `WARNING`, `LAR_FAILED`, `INFO_AVAIL`, `PARTIAL_TESTED`, `CONF_FAIL`, `EMERG_OKAY`.

Option –m

If the `–m` option is specified, information is displayed on the assignment of MBPs on the SP bus to MBPs on the MBII bus. A line is generated for each MBP on the SP bus. A line is structured as follows:

```
bus : spcab : spslot : spchan : sp_ctlrnam; bus : mb2cab : mb2bus \ □
: mb2slot : mb2_ctlrnam : logid
```

`bus` Type of the relevant bus (SP, MBII).

`spcab`
Number of the system cabinet.

`spslot`
Number of the slot on the SP bus.

`spchan`
Channel number of the MBP on the SP bus.

`sp_ctrlnam`
Product name of the relevant MBP on the SP bus.

`mb2cab`
Number of the cabinet containing the MBII.

`mb2bus`
Number of the MBII bus.

`mb2slot`
Number of the slot on the MBII bus.

`mb2_ctrlnam`
Product name of the MBP on the MBII.

`logid`
Logical number of the MBP.

Option –h

If the `–h` option is specified, the RM600-E EHIOS boards with the connected (directly or over a SCI cable) bus systems are displayed with their status.

A distinction is made between the following connection types:

`ONBOARD`
Sub-bus (X86) on the EHIOS board.

`LOCAL`
IO cage directly connected to the EHIOS.

`SCI_LINK`
LC connector on the EHIOS board with the position identifier `lcnr` (0 corresponds to left or above, 1 is right or below).

`REMOTE`
IO cage connected over a cable by means of an LC connector.

All IO cages with the keyword `REMOTE` that can be accessed via this SCI ring are displayed below a `SCI_LINK` line; this means, in particular, that the IO cages that are connected by means of two different SCI rings will also appear twice in the output.

Option –M

If the –M option is specified, information on the memory configuration of the system is displayed.

The display is different on the RM600-xxx (MMM and CPM board) to that on the RM600-E machines (CP4 boards).

A line is generated for each CPM and MMM board with the following structure:

```
ctrl_name : cab : slot : agent : ib: start_addr : end_addr : actsize \ □
: size : bank_state
```

`ctrl_name`
: Controller product name.

`cab` Logical cabinet number.

`slot` Number of the slot on the bus.

`agent`
: Agent number of the controller on the bus.

`ib` Interleaving block. The field specifies whether individual memories operate "interleaved", and if so, how. If a memory does not operate "interleaved", this field remains empty. Otherwise, the running number in the interleaving chain is specified.

`start_addr`
: Start address of the memory in hexadecimal format. Corresponding "interleaved" memories have the same start address.

`end_addr`
: End address of the memory in hexadecimal format.

`act_size`
: Current memory size in megabytes. The size of the currently available memory is specified in this field (part of the memory bank may be defective or unusable).

`size` Maximum memory size in megabytes. The size of the available memory is specified in this field, provided all assembled memory banks are operational. (A memory module can have different capacities, depending on the equipment).

`bank_state`
: Indication of the bank status (defective, operational, unusable). One of the three characters "0,1,*" is output for each assembled bank on the board:

  `0` Defective bank.

  `1` Bank operational and used.

  `*` Bank operational but unusable.

  Examples:

  1100 Banks 0 and 1 are operational; banks 2 and 3 are defective.

  110* Banks 0, 1, and 3 are operational; bank 2 is defective. Bank 3 cannot be used, due to HW/FW limitations (since bank 2 is defective).

  11 Banks 0 and 1 are operational.

Lines are sorted firstly by the field `start_addr`, and then by the field `ib`.

A line with the following structure is created for each CP4 board:

```
ctrl_name: cab : slot : agent : start_addr : end_addr: actsize: size \ □
: mem_module_state : mem_module_size
```

`ctrl_name`
: Controller product name.

`cab` Logical cabinet number.

`slot` Number of the slot on the bus.

`agent`
: Agent number of the controller on the bus.

`start_addr`
: Start address of the memory in hexadecimal format.

`end_addr`
: End address of the memory in hexadecimal format.

`act_size`
: Current memory size in megabytes. The size of the currently available memory is specified in this field (some of the memory modules may be defective or unusable).

`size` Maximum memory size in megabytes. The size of the available memory is specified in this field, provided all assembled memory modules are operational. (A memory module can have different capacities, depending on the equipment).

`mem_module_state`
: Status of the memory modules (defective, operational, not assembled).

  A bank is distributed to each of two memory modules on the CP4 board. As a result, it may not be possible to use a module, even though it is operational.

  The display reads from left to right in ascending order of RAM slot numbers.

  One of the characters "0,1,–" is output for each RAM slot:

  – Module not assembled.

  `0` Defective module.

  `1` Module operational and in use.

  A blank is output after every 8 characters in order to improve clarity.

  The modules are output from left to right in ascending order of RAM slot numbers.

  Example:

  ```
  11111111 11110011 11-11111 -------
  ```

  Modules 0-11,14-17, 19-23 are operational.

  Modules 12 and 13 are defective.

  Modules 18, 24-31 are not assembled.

`mem_module_size`
: Size of the individual modules.

  A character is output for each module that indicates the size of the module:

  – Size unknown (module is possibly not assembled).

  `0` 16 MB

  `1` 32 MB

  `2` 64 MB

  `3` 128 MB

  A blank is output after every 8 characters in order to improve clarity.

The lines are sorted following the `start_addr` field.

DIAGNOSTICS

`showconf` ends with an exit code other than 0 in the event of errors. Error messages are self-explanatory.

SEE ALSO

`controller(4)`, `ldcontroller(8)`, `mkcontroller(8)`, `modcontroller(8)`.

System Administrator's Reference Manual

spbuserrlog(8)
RM600 only

NAME

`spbuserrlog -` provide hardware information after an SPbus error

SYNOPSIS

`spbuserrlog` [`–V`] *agent_type agent_id*

DESCRIPTION

`spbuserrlog` outputs hardware information for an SPbus agent to stdout after an SPbus error has occurred.

`spbuserrlog` requests the *agent_type* and *agent_id* parameters. *agent_type* can be one of the parameters `CPU`, `CPM`, `CP4`, `HIOS` or `EHIOS`.

Input for *agent_id* can be determined with the `showconf(8)` command under the "`ag`" column.

`spbuserrlog` only returns hardware information if the agent has also stored error information.

`spbuserrlog` requires superuser rights.

OPTIONS

`–V` Outputs the version number.

RETURN VALUES

`spbuserrlog` returns `0` if the command was executed successfully, or otherwise:

`12` Main memory full.

`22` Invalid argument.

`61` No data available.

DIAGNOSTICS

`spbuserrlog: Invalid error log or action invalid for agent type` *agent_type*
The agent has not stored error information or an incorrect parameter was specified.

EXAMPLES

The command:

```
spbuserrlog CP4 0
```

returns the following information, for example:

```
------------------------------ CPU Registers ---------------- 21.01.97  11:21:41
General Registers:
at_01 BFC90000   v0_02 F2AAD432  v1_03 BFD20000  a0_04 000003D0   a1_05 BFE08006
a2_06 BFE08007   a3_07 BFC8FF20  t0_08 00000001  t1_09 000000CF   t2_10 00000000
t3_11 BFC04DA4   t4_12 00000008  t5_13 00000001  t6_14 000003D0   t7_15 00000064
t8_24 00017D40   t9_25 F2AC1F2C  s0_16 00000B43  s1_17 BFC8C020   s2_18 81472500
......
```

SEE ALSO

`showconf(8)`, `spbusevent(8)`.

System Administrator's Reference Manual

spbusevent(8)
RM600 only

NAME

`spbusevent` - determine whether an SPbus error has occurred

SYNOPSIS

`spbusevent` [`–g`] [`–c`] [`–q`] [`–V`] *agent_type agent_id*

DESCRIPTION

`spbusevent` determines whether an SPbus error has occurred at an agent, and sends the result to stdout. The following status is a possible result: An SPbus error did or did not occur. The status can be reset with the `–c` option.

This command can only be used properly in conjunction with the `spbuserrlog(8)` command.

`spbusevent` requests the *agent_type* and *agent_id* parameters. *agent_type* can be one of the parameters `CPU`, `CPM`, `CP4`, `HIOS` or `EHIOS`.

Input for *agent_id* can be determined with the `showconf(8)` command under the "`ag`" column.

`spbusevent` requires superuser rights.

`spbusevent` can only be used on the RM600-E, or otherwise an error message is issued (see **DIAGNOSTICS**).

OPTIONS

No option corresponds to the `–g` option.

`–g` Determines the status of the SPbus error. Two messages are possible:

"`No SPbus error seen at agent` *n*" = No error has occurred.

"`SPbus error seen at agent` *n*" = An error has occurred.

`–c` Resets the status of the SPbus error to the status "no error". When called again with the `–g` option, a message is issued to the effect that there is no error:

"`SPbus Error Flag cleared at agent` *n*"

`–q` Suppresses output to stdout.

`–V` Outputs the version number.

RETURN VALUES

`spbusevent` returns `0` if the command was executed successfully, or otherwise:

`12` Main memory full.

`22` Invalid argument.

`61` No data available.

`240` No SPbus error determined.

`241` SPbus error determined.

DIAGNOSTICS

`spbusevent: Invalid error log or action invalid for agent type`
agent_type The agent has not stored error information or an incorrect parameter has been specified.

`service on this hardware not available`
The command was not called on an RM600-E.

EXAMPLES

The command:

```
spbusevent –g CP4 0
```

returns the following information, for example:

"SPbus error seen at agent 0"

SEE ALSO

showconf(8), spbuserrlog(8).

System Administrator's Reference Manual

sshd(8)

NAME

sshd - secure shell daemon

SYNOPSIS

sshd [–b *bits*] [–d] [–f *config_file*] [–g *login_grace_time*] [–h *host_key_file*] [–i] [–k *key_gen_time*] [–p *port*] [–q]

DESCRIPTION

sshd (Secure Shell Daemon) is the daemon program for ssh(1). Together these programs replace rlogin and rsh programs and provide secure encrypted communication between two untrusted hosts over an insecure network. The programs are intended to be as easy to install and use as possible.

sshd is the daemon that listens for connections from clients. It is normally started at boot from /etc/rc.local or equivalent. It forks a new daemon for each incoming connection. The forked daemons handle key exchange, encryption, authentication, command execution and data exchange.

sshd works as follows. Each host has a host-specific RSA (Rivest-Shamir-Adelman) key (normally 1024 bits) used to identify the host. Additionally, when the daemon starts, it generates a server RSA key (normally 768 bits). This key is normally regenerated every hour if it has been used and is never stored on disk.

Whenever a client connects the daemon, the daemon sends its host and server public keys to the client. The client compares the host key against its own database to verify that it has not changed. The client then generates a 256 bit random number. It encrypts this random number using both the host key and the server key and sends the encrypted number to the server. Both sides then start to use this random number as a session key, which is used to encrypt all further communication in the session. The rest of the session is encrypted using a conventional cipher. Currently, Blowfish, IDEA, DES, 3DES are supported; 3DES is used by default (for detailed information see http://www.ssh.com/tech/crypto/algorithms.html). The client selects the encryption algorithm to use from those offered by the server.

Next, the server and the client enter an authentication dialog. The client tries to authenticate itself using .rhosts authentication, .rhosts authentication combined with RSA host authentication, RSA challenge-response authentication or password-based authentication.

.rhosts authentication is normally disabled because it is fundamentally insecure, but can be enabled in the server configuration file if desired. System security is not improved unless rshd(1M), rlogind(1M) and rexecd(1M) are disabled (thus completely disabling rlogin(1) and rsh(1) into that machine).

If the client successfully authenticates itself, a dialog for preparing the session is entered. At this time, the client may request things like allocating a pseudo-tty, forwarding X11 connections, forwarding TCP/IP connections or forwarding the authentication agent connection over the secure channel.

Finally, the client either requests a shell or execution of a command. The sides then enter session mode. In this mode, either side may send data at any time and such data is forwarded to/from the shell or command on the server side and the user terminal on the client side.

When the user program terminates and all forwarded X11 and other connections have been closed, the server sends the command exit status to the client and both sides exit.

sshd can be configured using command-line options or a configuration file. Command-line options override values specified in the configuration file.

OPTIONS

–b *bits*
: Specifies the number of bits in the server key (default 768).

–d
: Debug mode. The server sends verbose debug output to the system log, and does not put itself in the background. The server also will not fork and will only process one connection. This option is only intended for debugging for the server.

`-f` *config_file*
: Specifies the name of the configuration file. The default is `/etc/sshd_config`.

`-g` *login_grace_time*
: Gives the grace time for clients to authenticate themselves (default 300 seconds). If the client fails to authenticate the user within this many seconds, the server disconnects and exits. A value of zero indicates no limit.

`-h` *host_key_file*
: Specifies the file from which the host key is read (default `/etc/ssh_host_key`). This option must be given if `sshd` is not run as root (as the normal host file is normally not readable by anyone but root).

`-i`
: Specifies that `sshd` is being run from `inetd`(1M). `sshd` is normally not run from `inetd` because it needs to generate the server key before it can respond to the client and this may take tens of seconds. Clients would have to wait too long if the key were regenerated every time. With small key sizes (e.g. 512), however, it may be feasible to use `sshd` from `inetd`.

`-k` *key_gen_time*
: Specifies how often the server key is regenerated (default 3600 seconds, or one hour). The motivation for regenerating the key fairly often is that the key is not stored anywhere and, after about an hour, it becomes impossible to recover the key for decrypting intercepted communication even if the machine is cracked into or physically seized. A value of zero indicates that the key will never be regenerated.

`-p` *port*
: Specifies the port on which the server listens for connections (default 22).

`-q`
: Quiet mode. Nothing is sent to the system log. Normally the beginning, authentication and termination of each connection is logged.

CONFIGURATION FILE

`sshd` reads configuration data from `/etc/sshd_config` (or the file specified with `-f` at the command line). The file contains keyword-value pairs, one per line. Lines starting with "#" and empty lines are interpreted as comments.

The following keywords are possible.

`AFSTokenPassing`
: Specifies whether to accept AFS tokens passed from the client. Default is "`yes`".

`AllowHosts`
: This keyword can be followed by any number of host name patterns, separated by spaces. If specified, login is allowed only from hosts whose name matches one of the patterns. "*" and "?" can be used as wildcards in the patterns. Normal name servers are used to map the client's host into a canonical host name. If the name cannot be mapped, its IP address is used as the host name. By default all hosts are allowed to connect.

  Note that `sshd` can also be configured to use tcp wrappers using the `--with-libwrap` compile time configuration option.

`DenyHosts`
: This keyword can be followed by any number of host name patterns, separated by spaces. If specified, login is disallowed from the hosts whose names match any of the patterns.

`FascistLogging`
: Specifies whether to use verbose logging. Verbose logging violates the privacy of users and is not recommended. The argument must be "`yes`" or "`no`" (without the quotes). The default is "`no`".

`HostKey`
: Specifies the file containing the private host key (default `/etc/ssh_host_key`).

`IgnoreRhosts`
: Specifies that `.rhosts` and `.shosts` files will not be used in authentication. `/etc/hosts.equiv` and `/etc/shosts.equiv` are still used. The default is "`no`".

`KeepAlive`

Specifies whether the system should send keepalive messages to the other side. If they are sent, the death of the connection or crash of one of the machines will be properly noticed. However, this means that connections will die if the route is down temporarily and some people find this annoying. On the other hand, if keepalives are not sent, sessions may hang indefinitely on the server, leaving "ghost" users and consuming server resources.

The default is "yes" (to send keepalives) and the server will notice if the network goes down or the client host reboots. This avoids infinitely hanging sessions.

To disable keepalives, the value should be set to "no" in both the server and the client configuration files.

`KerberosAuthentication`
Specifies whether Kerberos authentication is allowed. This can be in the form of a Kerberos ticket or, if `PasswordAuthentication` is "yes", the password provided by the user will be validated through the Kerberos KDC/AFS kaserver/DCE Security Server. Default is "yes".

`KerberosOrLocalPasswd`
If set and if password authentication through Kerberos fails then the password will be validated via any additional local mechanism, such as `/etc/passwd` or security ID. Default is "no".

`KerberosTgtPassing`
Specifies whether a Kerberos TGT may be forwarded to the server. Default is "no", TGT forwarding only works with the AFS kaserver.

`KerberosTicketCleanup`
Specifies whether to automatically destroy the user's ticket cache file on logout. Default is "yes".

`KeyRegenerationInterval`
The server key is automatically regenerated after this many seconds (if it has been used). The purpose of regeneration is to prevent decrypting captured sessions by later breaking into the machine and stealing the keys. The key is never stored anywhere. If the value is 0, the key is never regenerated. The default is 3600 (seconds).

`LoginGraceTime`
The server disconnects after this time if the user has not successfully logged in. If the value is 0, there is no time limit. The default is 600 (seconds).

`PasswordAuthentication`
Specifies whether password authentication is allowed. The default is "yes".

`PermitEmptyPasswords`
When password authentication is allowed, it specifies whether the server allows login to accounts with empty password strings. The default is "yes".

`PermitRootLogin`
Specifies whether the root can log in using `ssh(1)` . The default is "yes".

Root login with RSA authentication when the *command* option has been specified will be allowed regardless of the value of this setting (which may be useful for taking remote backups even if root login is normally not allowed).

`Port` Specifies the port number that `sshd` listens on. The default is 22.

`PrintMotd`
Specifies whether `sshd` should print `/etc/motd` when a user logs in interactively. (On some systems it is also printed by the shell, `/etc/profile`, or equivalent.) The default is "yes".

`QuietMode`
Specifies whether the system runs in quiet mode. In quiet mode, nothing is logged in the system log, except fatal errors. The default is "no".

`RandomSeed`
Specifies the file containing the random seed for the server; this file is created automatically and updated regularly. The default is `/etc/ssh_random_seed`.

`RhostsAuthentication`
Specifies whether authentication using `.rhosts` or `/etc/hosts.equiv` files is sufficient. Normally,

this method should not be permitted because it is insecure. `RhostsRSAAuthentication` should be used instead, because it performs RSA-based host authentication in addition to normal `.rhosts` or `/etc/hosts.equiv` authentication. The default is "`no`".

`RhostsRSAAuthentication`
Specifies whether `.rhosts` or `/etc/hosts.equiv` authentication together with successful RSA host authentication is allowed. The default is "`yes`".

`RSAAuthentication`
Specifies whether pure RSA authentication is allowed. The default is "`yes`".

`ServerKeyBits`
Defines the number of bits in the server key. The minimum value is 512 and the default is 768.

`StrictModes`
Specifies whether `ssh` should check file modes and ownership of the user's home directory and `.rhosts` files before accepting login. This is normally desirable because novices sometimes accidentally leave their directory or files world-writeable. The default is "`yes`".

`SyslogFacility`
Gives the facility code that is used when logging messages from `sshd`. The possible values are: `DAEMON`, `USER`, `AUTH`, `LOCAL0`, `LOCAL1`, `LOCAL2`, `LOCAL3`, `LOCAL4`, `LOCAL5`, `LOCAL6`, `LOCAL7`. The default is `DAEMON`.

`X11Forwarding`
Specifies whether X11 forwarding is permitted. The default is "`yes`". Note that disabling X11 forwarding does not improve security in any way, as users can always install their own forwarders.

LOGIN PROCESS

When a user successfully logs in, `sshd` does the following:

1. If the login is on a tty, and no command has been specified, prints the last login time and `/etc/motd` (unless prevented in the configuration file or by `$HOME/.hushlogin`; see the **FILES** section).
2. If the login is on a tty, records login time.
3. Checks `/etc/nologin`; if it exists, prints contents and quits (unless root).
4. Changes to run with normal user privileges.
5. Sets up basic environment.
6. Reads `/etc/environment`, if it exists.
7. Reads `$HOME/.ssh/environment`, if it exists.
8. Changes to user's home directory.
9. If `$HOME/.ssh/rc` exists, runs it; otherwise if `/etc/sshrc` exists, runs it; otherwise runs `xauth`. The "rc" files are given the X11 authentication protocol and cookie in standard input.
10. Runs user's shell or command.

AUTHORIZED_KEYS FILE FORMAT

The `$HOME/.ssh/authorized_keys` file lists the RSA keys that are permitted for RSA authentication. Each line of the file contains one key (empty lines and lines starting with a "#" are ignored as comments). Each line consists of the following fields, separated by spaces: options, bits, exponent, modulus, comment. The options field is optional; its presence is determined by whether the line starts with a number or not (the option field never starts with a number). The bits, exponent, modulus and comment fields give the RSA key; the comment field is not used for anything (but may be convenient for the user to identify the key).

Note that lines in this file are usually several hundred bytes long (because of the size of the RSA key modulus). You don't want to type them in; instead, copy the `identity.pub` file and edit it.

The options (if present) consists of comma-separated option specifications. No spaces are permitted, except within double quotes. The following option specifications are supported:

`from="pattern-list"`

Specifies that in addition to RSA authentication, the canonical name of the remote host must be present in the comma-separated list of patterns ("*" and "?" serve as wildcards). The list may also contain patterns negated by prefixing them with "!"; if the canonical host name matches a negated pattern, the key is not accepted. The purpose of this option is to optionally increase security: RSA authentication by itself does not trust the network or name servers or anything (but the key); however, if somebody somehow steals the key, the key permits an intruder to log in from anywhere in the world. This additional option makes using a stolen key more difficult (name servers and/or routers would have to be compromised in addition to just the key).

`command="command"`

Specifies that the command is executed whenever this key is used for authentication. The command supplied by the user (if any) is ignored. The command is run on a pseudo-tty if the connection requests a pseudo-tty; otherwise it is run without a tty. A quote may be included in the command by quoting it with a backslash. This option might be useful to restrict certain RSA keys to perform just a specific operation. An example might be a key that permits remote backups but nothing else. Notice that the client may specify TCP/IP and/or X11 forwardings unless they are explicitly prohibited.

`environment="NAME=value"`

Specifies that the string is to be added to the environment when logging in using this key. Environment variables set this way override other default environment values. Multiple options of this type are permitted.

`no-port-forwarding`
: Forbids TCP/IP forwarding when this key is used for authentication. Any port forward requests by the client will return an error. This might be used, for example, in connection with the *command* option.

`no-X11-forwarding`
: Forbids X11 forwarding when this key is used for authentication. Any X11 forward requests by the client will return an error.

`no-agent-forwarding`
: Forbids authentication agent forwarding when this key is used for authentication.

`no-pty`
: Prevents tty allocation (a request to allocate a pseudo-tty will fail).

Examples

```
1024 33 12121...312314325 ylo@foo.bar
from="*.niksula.hut.fi,!pc.niksula.hut.fi" 1024 35 23...2334 ylo@niksula
command="dump /home",no-pty,no-port-forwarding 1024 33 23...2323 backup.hut.fi
```

SSH_KNOWN_HOSTS FILE FORMAT

The `/etc/ssh_known_hosts` and `$HOME/.ssh/known_hosts` files contain host public keys for all known hosts. The global file should be prepared by the administrator (optional); the per-user file is maintained automatically. Whenever the user connects an unknown host, its key is added to the per-user file.

Each line in these files contains the following fields: hostnames, bits, exponent, modulus, comment. The fields are separated by spaces.

Hostnames is a comma-separated list of patterns ("*" and "?" act as wildcards); each pattern in turn is matched against the canonical host name (when authenticating a client) or against the user-supplied name (when authenticating a server). A pattern may also be preceded by "!" to indicate negation: if the host name matches a negated pattern, it is not accepted (by that line) even if it matched another pattern on the line.

Bits, exponent, and modulus are taken directly from the host key; they can be obtained, for example, from `/etc/ssh_host_key.pub`. The optional comment field continues to the end of the line and is not used.

Lines starting with "#" and empty lines are ignored as comments.

When performing host authentication, authentication is accepted if any matching line has the proper key. It is thus permissible (but not recommended) to have several lines or different host keys for the same names. This will inevitably happen when short forms of host names from different domains are put in the file. It is possible that the files contain conflicting information; authentication is accepted if valid information can be found from either file.

Note that the lines in these files are typically hundreds of characters long and you definitely don't want to type in the host keys by hand. Rather, generate them with a script or by taking `/etc/ssh_host_key.pub` and adding the host names at the front.

Examples

```
closenet,closenet.hut.fi,...,130.233.208.41 1024 37 159...93 closenet.hut.fi
```

FILES

`/etc/sshd_config`
: Contains configuration data for `sshd`. This file should be writeable by root only, but it is recommended (though not necessary) that it be world-readable.

`/etc/ssh_host_key`
: Contains the private part of the host key. This file is normally created automatically by "make install", but can also be created manually using `ssh-keygen(1)`. This file should only be owned by root, readable only by root and not accessible to others.

`/etc/ssh_host_key.pub`
: Contains the public part of the host key. This file is normally created automatically by "make install", but can also be created manually. This file should be world-readable but writeable only by root. Its contents should match the private part. This file is not really used for anything; it is only provided for the convenience of the user so its contents can be copied to known hosts files.

`/etc/ssh_random_seed`
: This file contains a seed for the random number generator. This file should only be accessible by root.

`/etc/sshd.pid`
: Contains the process id of the `sshd` listening for connections (if there are several daemons running concurrently for different ports, this contains the pid of the one started last). The contents of this file are not sensitive; it can be world-readable.

`$HOME/.ssh/authorized_keys`
: Lists the RSA keys that can be used to log into the user's account. This file must be readable by root (which may on some machines imply it being world-readable if the user's home directory resides on an NFS volume). It is recommended that it not be accessible by others. The format of this file is described above.

`/etc/ssh_known_hosts`
`$HOME/.ssh/known_hosts`
: These files are consulted when using `.rhosts` with RSA host authentication to check the public key of the host. The key must be listed in one of these files to be accepted. (The client uses the same files to verify that the remote host is the one we intended to connect.) These files should be writeable only by root/the owner. `/etc/ssh_known_hosts` should be world-readable and `$HOME/.ssh/known_hosts` can but need not be world-readable.

`/etc/nologin`
: If this file exists, `sshd` refuses to let anyone except root log in. The contents of the file are displayed to anyone trying to log in and non-root connections are refused. The file should be world-readable.

`$HOME/.rhosts`
: This file contains host-username pairs, separated by a space, one per line. The given user on the corresponding host is permitted to log in without password. The same file is used by `rlogind` and `rshd`. `ssh` differs from `rlogind` and `rshd` in that it requires RSA host authentication in addition to validating the host name retrieved from domain name servers (unless compiled with the `--with-rhosts` configuration option). The file must be writeable only by the user; it is recommended that it not be accessible by others.

  If is also possible to use netgroups in the file. Either host or user name may be of the form `+@groupname` to specify all hosts or all users in the group.

`$HOME/.shosts`
: For `ssh`, this file is exactly the same as for `.rhosts`. However, this file is not used by `rlogin` and

rshd, so using this permits access using ssh only.

/etc/hosts.equiv
: This file is used during .rhosts authentication. In the simplest form, this file contains host names, one per line. Users on these hosts are permitted to log in without a password, provided they have the same user name on both machines. The host name may also be followed by a user name; such users are permitted to log in as *any* user on this machine (except root). Additionally, the syntax +@group can be used to specify netgroups. Negated entries start with "–".

  If the client host/user is successfully matched in this file, login is automatically permitted provided the client and server user names are the same. Additionally, successful RSA host authentication is normally required. This file must be writeable only by root; it is recommended that it be world-readable.

  Warning: It is almost never a good idea to use user names in hosts.equiv. Beware that it really means that the named user(s) can log in as *anybody*, which includes bin, daemon, adm and other accounts that own critical binaries and directories. Using a user name practically grants the user root access. The only valid use for user names that I can think of is in negative entries. Note that this warning also applies to rsh/rlogin.

/etc/shosts.equiv
: This is processed exactly as /etc/hosts.equiv. However, this file may be useful in environments that want to run both rsh/rlogin and ssh.

/etc/environment
: This file is read into the environment at login (if it exists). It can only contain empty lines, comment lines (that start with "#") and assignment lines in the form name=value. This file is processed in all environments (normal rsh/rlogin only processes it on AIX and potentially some other systems). The file should be writeable only by root, and should be world-readable.

$HOME/.ssh/environment
: This file is read into the environment after /etc/environment. It has the same format. The file should be writeable only by the user; it need not be readable by anyone else.

$HOME/.ssh/rc
: If this file exists, it is run with /bin/sh after reading the environment files but before starting the user's shell or command. If X11 spoofing is used, this will receive the "proto cookie" pair in standard input (and DISPLAY in environment). This must call xauth in that case.

  The primary purpose of this file is to run any initialization routines that may be needed before the user's home directory becomes accessible; AFS is a particular example of such an environment.

  This file will probably contain some initialization code followed by something similar to: "`if read proto cookie; then echo add $DISPLAY $proto $cookie | xauth –q –; fi`".

  If this file does not exist, /etc/sshrc is run and if that does not exist either, xauth is used to store the cookie.

  This file should be writeable only by the user and need not be readable by anyone else.

/etc/sshrc
: Like $HOME/.ssh/rc. This can be used to specify machine-specific login-time initializations globally. This file should be writeable only by root and should be world-readable.

INSTALLATION

sshd is normally run as root. If it is not run as root, it can only log in as the user it is running as and password authentication may not work if the system uses shadow passwords. An alternative host key file must also be used.

sshd is normally started from /etc/rc.local or equivalent at system boot.

REFERENCES

http://www.ssh.com

http://www.ssh.org

SEE ALSO

rlogin(1), rsh(1), scp(1), ssh(1), ssh-add(1), ssh-agent(1), ssh-keygen(1).

System Administrator's Reference Manual

statis(8)

NAME

`statis -` manage kernel statistics gathering

SYNOPSIS

`statis` [`–y` *facility*] [`–n` *facility*] [`–c` *facility*] [`–p` *facility*] [`–t`]

DESCRIPTION

`statis` manages the collection of various kernel performance monitoring statistics. Collection of kernel statistics has associated costs. In order to minimize the effects of performance monitoring, the `statis` command may be used to disable some of the statistics collection.

OPTIONS

`–y` *facility*
: Enables statistics gathering for *facility*.

`–n` *facility*
: Disables statistics gathering for *facility*.

`–c` *facility*
: Zeroes the *facility* statistics.

`–p` *facility*
: Displays the *facility* statistics in a raw dump format. Other system status display programs provide easier to use formats.

`–t`
: Displays all of the facilities, one per line.

DIAGNOSTICS

Exit status is normally `0`. If an error occurs, exit status is `1`.

NOTES

Some statistic facilities cannot be disabled or cleared.

Statistic facilities that are already activated by default cannot be activated again, rather can be displayed with the `–p` option.

Statistic facilities that by default must always be available cannot be deactivated.

SEE ALSO

`sar`(1), `mpstat`(1M), `vmstat`(1M).

System Administrator's Reference Manual

switch(8)

NAME

`switch -` use a peripheral switch

SYNOPSIS

`/etc/opt/havail/switch` *tty_dev* [`–v`] [`–k` *kask*] [`–a` *addr*] {`–c` *chn* | `–s`}

`/etc/opt/havail/switch` *tty_dev* [`–v`] [`–k` *kask*] [`–a` *addr*] `–c` *chn* `–s`

DESCRIPTION

The `switch` allows software-dependent switching of the integrated SCSI switch for all disks in the peripheral cabinet. This switch enables two different systems to connect to the external SCSI disks, after which they can then be switched to the relevant system as required. In conjunction with the virtual disk unit, this SCSI switch allows the creation of a standby configuration in which one or more external disks are connected to two systems. The SCSI switch then switches one of the two systems to the external disks.

The `switch` command allows you to switch from SCSI, as from terminal peripherals.

tty_dev specifies the device file which controls the switch. Instead of specifying the data lines to be switched, the *tty_dev* file specifies the device file of an SS97 or V24 interface of the system to which the control line of the switch is connected.

OPTIONS

The following options are available:

`–a` *addr*
: The address (number) of the module to be switched. In the case of an integrated SCSI switch, at present only the values `1` or `a` are practical as they have the same meaning. The selection of either will result in a switch.

`–k` *kask*
: The `–k` option specifies the number of the integrated SCSI switch to be switched, if more than one of this type of switch is to be switched at once (cascading). Possible values for *kask* are, a value from `1` to `9`, or `a` if all switches are specified. `a` can only be specified in conjunction with the `–s` option. If this option is not specified, the first SCSI switch must be switched.

`–c` *chn*
: The `–c` option specifies the channel to be set. Possible entries for *chn* are `1` or `2` (alternatively, also `A` or `B`).

`–s`
: The `–s` queries the current switch setting. In this case, `switch` outputs the current channel setting (1 or 2) as standard output. `–s` must always be the last option specified.

NOTES

When the disk is removed from a system currently running `switch`, this normally has unpleasant results, particularly for the file system. Switching is permitted in emergency situations only, i.e. if a system in the standby configuration fails, or if all applications are ended and all files systems are unmounted.

EXAMPLES

Setting the firist peripheral cabinet to channel 2:

```
switch /dev/term/15 –c 2
```

Query all switches of the cascaded cabinets:

```
switch /dev/term/15 –k a –s
```

DIAGNOSTICS

On its return, the `switch` returns an end status, which can be checked. The following values are possible:

`0` = `switch` ran without any errors.

`1` = status of the switch cannot be determined.

`3` = status following switching could not be determined or does not correspond to the estimated status.

`255` = an error occurred when `switch` was executing.

SEE ALSO

`changedev`(8), `reinitdev`(8), `switchscsi`(8), `switchsusi`(8).

System Administrator's Reference Manual

switchscsi(8)
RM400 only

NAME

`switchscsi -` operate a SCSI peripheral cabinet

SYNOPSIS

`/etc/opt/havail/switchscsi` [`–k` *kask*] `–c` *chn device*

`/etc/opt/havail/switchscsi` [`–k` *kask*] `–s` *device*

DESCRIPTION

The `switchscsi` command is used (on the software side) to switch the integrated SCSI switch in an external device cabinet. There is a SCSI connection from every two systems to the external cabinet. Each of the two systems is also connected to the device cabinet via a control line, which is used to send the signals for controlling the switch.

device indicates the device file belonging to the control line.

OPTIONS

The following options are available:

`–k` *kask*
: *kask*, a digit between 1 and 9, indicates the device cabinet to be controlled. *kask*=1 is set by default.

`–c` *chn*
: If `–c` is set, switching occurs. The *chn* argument is `1` or `2`, depending on which channel switching is to occur.

`–s`
: If `–s` is set, status information is output.

NOTES

If disks are switched on a system during operation with `switchscsi`, this usually has undesirable consequences, particularly for the file systems. Switching is only permitted in the case of an emergency, for example if a system has failed in a standby configuration, or if all applications have ended and the file systems have been unmounted.

EXAMPLES

Setting the third cascaded cabinet to channel 2:

```
switchscsi –k3 –c2 /dev/term/05
```

Querying the status of the third cascaded cabinet:

```
switchscsi –a3 –s /dev/term/05
```

gives:

```
switchscsi: extension box is switched to channel 2 (= B)
```

DIAGNOSTICS

`0` = `switchscsi` ran without errors.

`1` = Incorrect call.

`2` = Error when opening the control line.

`3` = Error when reading from/writing to the control line.

`4` = Internal error.

SEE ALSO

`changedev`(8), `reinitdev`(8), `switchscu2`(8), `switchsusi`(8).

System Administrator's Reference Manual

switchscu2(8)

NAME

`switchscu2` - operate peripheral switch

SYNOPSIS

`/etc/opt/havail/switchscu2` [–v] [–p *pbay*] [–m *module*] {–c *chn* | –s}

`/etc/opt/havail/switchscu2` [–v] [–p *pbay*] [–m *module*] –c *chn* –s

DESCRIPTION

`switchscu2` allows a SCSI switch to be both queried and activated in peripherals cabinets with switchable SCSI channels. The query sends the current setting of the switch to standard output. It is also possible to query all switches simultaneously.

The `switchscu2` command name contains the identifier for the original switch. `scu2` stands for SCU2 (SCSI Conversion Unit 2). A further switch (SISO, SCSI Isolator) is now also available, which can likewise be activated with this command.

The SCU2 switch can be put in two settings. These are called `1` and `2`, or alternatively `A` and `B`. The default setting (when the switch is powered up) is `1`. Only one switch at a time can be activated with a command call.

The SISO switch is used for the same purpose, though it has a further setting, apart from the previous two given. This setting is visible before the switch is first activated after it has been queried. The standard setting for the switch (when the switch is powered up) is `3`. In this switch position, the SCSI peripheral is connected with both systems, otherwise referred to as a multihost configuration. The `switchscu2` command will only allow the switch to be modified if it is set to either 1 or 2.

The switches are used to alternately connect the devices of a SCSI channel to two systems, thus facilitating high availability.

OPTIONS

–v Verbose specification. The command says what `switchscu2` is currently doing.

–p *pbay*

The –p option determines the peripherals cabinet to be selected. *pbay* is an integer. The range of values permitted for this option can be found further below. The integer stands for a cabinet number as interpreted by the EIP (Environmental Interface Processor). Two types of EIP are currently being used, i.e. EIP2 and EIP3. These will be looked at more closely here since they are differentiated on the basis on how they are selected internally.

1) "pure-bred" configurations, i.□e. only

1.1) EIP2 in the system.

The peripherals cabinet number specified with the *pbay* argument is interpreted as a so-called logical cabinet number and the peripherals cabinet is selected by means of this number. *pbay* is an integer between `1` and `15` or the character `a` (all peripherals cabinets). If the option is not specified, the default value `1` applies for *pbay*.

Note:

The assignment of logical cabinet numbers to physical cabinet numbers is performed at the same time as the physical cabinet number is notified to the system. This is done using the EIP `sys` command.

1.2) EIP3 in the system.

The peripherals cabinet number specified with the *pbay* argument is interpreted as a physical cabinet number (i.□e. real cabinet number, as is set on the DIP switch of the peripherals cabinet) and the peripherals cabinet is selected by means of this number. *pbay* is an integer between `0` and `63`. If the *pbay* argument is not specified, the default

value `0` applies for *pbay*.

2) Mixed configurations, i.□e. systems with EIP2 and EIP3.

To enable a uniform cabinet number to be used in this type of system network both by a system with EIP2 and a system with EIP3, the cabinet numbers must be assigned as follows: logical number equals physical number. This limits the number of peripherals cabinets that can be selected by systems with EIP2 to a maximum of `15`.

Two examples are now given for clarification purposes:

2.1) System cabinet with EIP3, peripherals cabinet with EIP2 or EIP3.

The *pbay* number specified on this system with the `switchscu2` command is interpreted as a physical number.

Please refer to 2) for the permitted range of values and the default value.

2.2) System cabinet with EIP2, peripherals cabinet with EIP2 or EIP3.

The *pbay* number specified on this system with the `switchscu2` command is interpreted as a logical number.

The following must be strictly observed when configuring cabinet numbers in this type of mixed system:

- the logical number of this/these peripherals cabinet(s) has to match the physical number
- the number of peripherals cabinets is limited to `15` (maximum number of peripherals cabinets (EIPs), that can be selected by an EIP2)

Please refer to 1) for the permitted range of values and the default value.

The EIP type that will select the switch to be activated can be determined with the `ups`(1M) command.

`-m` *module*

module is an integer between `1` and `9` or the character `a` (all switches in the peripherals cabinet) and indicates the switches (SCSI channel) affected in the peripherals cabinet. If the `-m` option is not specified, the default `1` applies for *module*.

`-c` *chn*

The `-c` option indicates which channel is to be set. In this case, either `1` or `2` (or `A` or `B`) can be given as the channel name.

When the switch has been activated, the setting for the switch after the switching has occurred is sent to standard output.

When the `-s` option is used, the setting of the switch to be activated is output before the switch is actually activated.

`-s` The `-s` option is used to query the setting of one or more switches (see also the `-p` and `-m` option). In this case, `switchscu2` sends a notification to standard output as to which channel (1 or 2, or also channel 3 in the case of the SISO switch) of the switch is currently set.

EXAMPLES

Setting the shelf no. 1 of the peripherals cabinet no. 3 to channel 2; in this case the channel setting should be displayed before switching occurs:

```
switchscu2 -c 2 -p 3 -s
```

Querying all switches in the peripherals cabinet no. 1:

```
switchscu2 -m a -s
```

Querying all switches with no. 1 in all peripherals cabinets:

```
switchscu2 -p a -s
```

DIAGNOSTICS

The `switchscu2` program indicates a final status when it returns, and this can be checked. The following values are possible here:

`0` = `switchscu2` ran without errors.

`1` = The status of a switch cannot be determined.

`255` =
An error occurred while `switchscu2` was running.

SEE ALSO

`ups(1M)`, `eip2(7)`, `changedev(8)`, `reinitdev(8)`.

System Administrator's Reference Manual

switchsusi(8)

NAME

`switchsusi -` operate a SUSI peripheral switch

SYNOPSIS

`/etc/opt/havail/switchsusi` [`–a` *addr*] [`–k` *kask*] [`–s`] `–c` *chn device*

`/etc/opt/havail/switchsusi` [`–a` *addr*] [`–k` *kask*] `–s` *device*

DESCRIPTION

The `switchsusi` command is used to switch (on the software side) the serial devices (terminals, printers) connected to the SUSI (system UNIX/SINIX switch interface). From every two systems there is a serial line (V.24 or SS97) for each device connected to the SUSI, and also a corresponding line to the peripherals from the SUSI. The SUSI is also connected to each system via a control line, which is used to send signals to control the switch.

device indicates the device file belonging to the control line.

OPTIONS

The following options are available:

`–a` *addr*
: *addr*, a number between 1 and 10, indicates the module to be controlled; all modules if *addr*=A. *addr*=1 is set by default.

`–k` *kask*
: *kask*, a digit between 1 and 9, indicates the *kask*'th cascaded SUSI to be addressed. *kask*=1 is set by default.

`–c` *chn*
: If `–c` is set, switching occurs. The *chn* argument is `1` or `2`, depending on which channel switching is to occur.

`–s`
: If `–s` is set, status information is output.

EXAMPLES

Setting the third module to channel 2:

```
switchsusi –a3 –c2 /dev/term/06
```

Querying the status of the third module:

```
switchsusi –a3 –s /dev/term/06
```

gives:

```
switchsusi: module 3 is switched to channel 2 (= B)
```

DIAGNOSTICS

`0` = `switchsusi` ran without errors.

`1` = Incorrect call.

`2` = Error when opening the control line.

`3` = Error when reading from/writing to the control line.

`4` = Internal error.

`5` = Switching was not successful, or module is defective or does not exist.

SEE ALSO

`switchscsi`(8), `switchscu2`(8).

System Administrator's Reference Manual

tpsetup(8)

NAME

`tpsetup` - set up special files for a magnetic tape device

SYNOPSIS

`tpsetup` –`U` | –`C` *autoconf-name*

`tpsetup` –`a` *autoconf-name* [–`n` *symbolic-name*]

`tpsetup` –`d` *autoconf-name* | *symbolic-name*

`tpsetup` –`l`

DESCRIPTION

`tpsetup` is used to set up special files for a tape device in `/dev/ios0`. If the tape device has a serial number, a special file with a freely definable name can be created in `/dev/sym` in addition to the name in `/dev/ios0`. These so-called symbolic names are particularly useful for tape devices in cluster configurations [see `clos(7)`], as well as for tape devices connected to an FC/SCSI router [see `srout(7)`]. The symbolic names are managed globally for the cluster in cluster configurations

autoconf-name is the generic name of a tape device as displayed by `autoconf(8)` . For example, `ios0/stape014` is the SCSI tape drive with SCSI-ID 4 of the SCSI port 01.

symbolic-name is a freely definable name for the tape device; *symbolic-name* can also be a relative path name.

OPTIONS

–`U` All special files are created in `/dev/ios0` regardless of the features of the tape device.

–`C` In accordance with the features of the tape device, the required special files are created in `/dev/ios0` (i.e. only special files required for this tape device are created).

–`a` A special file with a symbolic name is set up in `/dev/sym` for the tape device specified by *autoconf-name*. `tpsetup` assigns the name `/dev/sym/tape`*x* by default, where *x* is the smallest decimal number that has not yet been used previously for another symbolic name. This name can be overridden with the additional –`n` option.

–`n` Overwrites the symbolic name assigned automatically with –`a` *autoconf-name*. Only one symbolic name is permitted for each tape device. If a symbolic name already exists for the selected tape device, it will be replaced by the new name.

–`d` The symbolic name for the tape device specified with *autoconf-name* or *symbolic-name* is deleted.

–`l` Lists all symbolic names currently assigned in the cluster.

SEE ALSO

`clos(7)` , `srout(7)` , `stape(7)` , `autoconf(8)` , `setinfo(8)` .

System Administrator's Reference Manual

upd_channel_id(8)
RM400 only

NAME

`upd_channel_id` - display/modify the channel IDs saved in the NVRAM

SYNOPSIS

`upd_channel_id` {`-l` | `-o` | [`-f`] `-c` | [`-f`] `-w` *file*}

DESCRIPTION

The channel ID is used to uniquely identify devices and is unique within its device-specific class. It forms the link between the software level (e.g. device name) and the slot.

At initial system startup, the device drivers are assigned a unique channel ID within the class. This configuration is stored in the NVRAM (non volatile random access memory).

Every time the system starts up, the channel ID is read by the device drivers. The entry in the NVRAM is read through specification of the slot information (e.g. `PCI#1/1`) and class (e.g. `ETHER`).

The saved configuration remains available if a device is no longer available, (e.g. due to repairs). If a device of the same type (the slot information must also be the same) is now attached to this slot, the saved configuration is used.

The saved configuration is lost if a device of a different class is attached with the same slot information.

The saved configuration can be read and modified with `upd_channel_id`.

Onboard entries can neither be displayed nor modified with the `upd_channel_id` command.

OPTIONS

`-l` Outputs a table containing all the channel IDs to the screen (stdout).

`-o` Used to convert (old) log IDs to (new) channel IDs for the `SCSI` class. As a rule, this is required once for the update installation. Outputs a table containing all the channel IDs to the screen (stdout). The channel IDs are calculated according to the old (`updeisa`) algorithm.

`-c` Deletes all channel ID entries from the NVRAM.

`-w` Reads the table of channel ID entries from the specified file, checks the information, and saves it in the NVRAM.

`-f` Deletes/writes to NVRAM for ROOT channels also.

file Contains the channel ID entries to be stored in the NVRAM.

A channel ID entry has the following structure:

`slot`[`.subchannel`], `channel_id`, [`class_code`], [`driver`], [`option`], [`comment`]

where

`slot` `EISA#1` | `EISA#2` | ... | `PCI#1` | ... | `PCI#14/14/14`

Interface type and slot number of the controller plug-in board.

`subchannel`

0 ... 9, for EISA 0 ... 7

`channel_id`

0 ... 99, for EISA log ID 0 ... 31

`class_code`

Device specific class.

`MISC` | `SCSI` | `FLOPPY` | `MASSSTORE` | `ETHER` | `TOKEN` | `FDDI` | `ATM` | `NETWORK` | `DISPLAY` | `MULTIMEDIA` | `COMMDEV`

The device specific classes are calculated on the basis of the PCI specification.

Entries using the old EISA log ID do not belong to any class.

`driver`
Name of the driver, e.g. `adp` or `fc`.

`option`
Optional parameter, used, for example, for a SCSI host ID.

`comment`
Brief controller description.

A channel ID entry based on the slot information (`slot` and `subchannel`), which must be unique within the list. The specified channel ID (`channel_id`) must also be unique within the specified class (`class_code`). The entries for the driver name (`driver`) and comment (`comment`) are only displayed and consequently cannot be set. The optional field (`option`) can be modified and is evaluated in relation to the specific driver.

For EISA log ID entries, the `option` field can neither be displayed nor set.

The `option` field has the following meaning for the `SCSI` class:

- The lower (right-hand) four bits contain the host adapter ID. This is preassigned a standard value by the corresponding driver (e.g. 0x07). Values greater than seven relate to a wide controller.
- Some host adapters support ULTRA SCSI. To enable this feature, bit 6 has to be set (e.g. 0x27 for ULTRA SCSI, host adapter ID 7).
- Some host adapters support ULTRA2 SCSI. The fifth bit is set in order to enable this feature to be used (e.g. 0x17 for ULTRA2 SCSI, host adapter ID 7).

  If the values for ULTRA SCSI and ULTRA2 SCSI are set in the `option` field, synchronous data transmission is only performed at a speed of 5 MHz (provided the host adapter supports this).
- If the `driver=fc` has the fifth bit set (0x10), the Pt-Pt protocol (point to point) is used as the FC protocol (fibre channel). This is necessary in order to connect an FC switch. The controller is indicated with "Top: PT-PT" in the output of `autoconf -l`.
- The seventh bit is set for multihost mode (e.g. 0x47). The controller is identified by the term "shared" in the `autoconf -l` output.
- In some special cases, where wide negotiations cause the device to malfunction, these wide negotiations are suppressed by setting the eight bit (e.g. 0x87).

EXAMPLES

Displaying the entries

The `upd_channel_id –l` command delivers the following output:

```
##slot.subchannel, channel_id, class_code, driver,   option, comment
#
EISA#1,               1,      ETHER,      mac596,    0x00,  ETHER SNI Intel 8259
EISA#2,               4,      SCSI,       adp,       0x07,  SCSI ADP ADP1740A DE
PCI#1/2,              0,      DISPLAY,    gfx,       0x00,  DISPLAY SNI CIRRUS 5
PCI#1/1,              2,      SCSI,       ncr,       0x07,  SCSI SNI SYM53C875 S
PCI#2,                3,      SCSI,       ,          0x48,
PCI#2.0,              5,      SCSI,       dpt,       0x07,  SCSI DPT PM3222 SE
PCI#2.1,              6,      SCSI,       dpt,       0x07,  SCSI DPT PM3222 SE
PCI#2.2,              7,      SCSI,       dpt,       0x07,  SCSI DPT PM3222 SE
PCI#3,                0,      TOKEN,      madge,     0x02,  TOKEN MADGE Smart 16
```

Slot		
EISA	1	SNI Ethernet controller
EISA	2	SCSI Adaptec AHA 1740A controller
PCI	1	Super Combo Board from SNI, with a graphics board and SCSI c jumper

PCI	2	A SCSI controller with the host adapter ID 8 was previously attac operated in multihost mode with wide peripherals.
PCI	2	3-channel SCSI controller from DPT
PCI	3	Token ring controller from Madge

Changing an entry

a) Display previous entries first:

```
# upd_channel_id -l
#
#slot.subchannel, channel_id, class_code, driver,   option, comment
#
EISA#1,               0,      ,           sih,       ,
PCI#1.0,              2,      SCSI,       dpt,       0x07,  SCSI DPT PM3224 SE,
PCI#2,                3,      SCSI,       fc,        0x00,  FC EMULEX LP6502-OS
```

b) Save previous display to a file:

```
# upd_channel_id -l >/tmp/file
```

c) Set flag 0x10 for fibre channel (fc):

```
# vi /tmp/file
PCI#2,                3,      SCSI,       fc,        0x10,  FC EMULEX LP6502-OS
```

d) Read modified file:

```
# upd_channel_id -w /tmp/file
```

e) Display entries again:

```
# upd_channel_id -l
#
# slot.subchannel, channel_id, class_code, driver,   option, comment
#
EISA#1,               0,      ,           sih,       ,
PCI#1.0,              2,      SCSI,       dpt,       0x07,  SCSI DPT PM3224 SE,
PCI#2,                3,      SCSI,       fc,        0x10,  FC EMULEX LP6502-OS
```

f) Perform a reboot so that the new settings are active.

NOTES

Changes made in the NVRAM with the `upd_channel_id` command are only effective after a reboot. The output (`upd_channel_id -l`) of the `driver` and `comment` fields will not be displayed correctly prior to this.

When the channel ID configuration has been changed, the current configuration is stored in the `/etc/info/channel_id` file when the system is powered down. The old configuration is stored in the `/etc/info/channel_id.old` file.

A binary map of the channel ID configuration (NVRAM) is copied to partition 10 of the root disk when the system is powered down.

FILES

`/dev/nvram`

SEE ALSO

`nvram(7)`, `sdisk(7)`, `stape(7)`, `autoconf(8)`.

System Administrator's Reference Manual

upd_node(8)
RM400 only

NAME

`upd_node` - check/correct device nodes for disks,RAID subsystems and tapes

SYNOPSIS

`/sbin/upd_node {[ -C ] | -c | -u}`

DESCRIPTION

`upd_node` checks and corrects the device nodes for disks, RAID subsystems and tapes under `/dev/ios0`. All devices displayed with the `autoconf -l` command have to be created as device nodes with the appropriate device class number (major) and device number (minor). This means that device nodes are not checked and corrected unless they are listed by `autoconf`.

The `/dev/ios1` directory is checked for reasons of compatibility. The `/dev/ios1` directory, if it exists, must be a symbolic link to `/dev/ios0`.

OPTIONS

`-C` All device nodes are checked. Incorrect device nodes are displayed. In the case of tape devices, only partition 0 is checked.

`-c` All device nodes are checked. Incorrect device nodes are displayed.

`-u` All incorrect or missing device nodes are corrected or created.

NOTES

In the case of tape devices, only partition 0 is checked. The features of tape devices and the installation of the respective device nodes are implemented by means of `tpsetup -C`.

FILES

```
/dev/ios0/*sdisk*
/dev/ios0/*sraid*
/dev/ios0/*stape*
```

SEE ALSO

`mknod`(1M), `devname`(7), `sdisk`(7), `stape`(7), `autoconf`(8), `tpsetup`(8), `upd_channel_id`(8)
.

System Administrator's Reference Manual

update(8)

NAME

`update` - update operating system

SYNOPSIS

`/etc/update –c[–o][–e][–i]`

`/etc/update –a[–o][–e][–i]`

`/etc/update –r` *file*

`/etc/update –h`

DESCRIPTION

The operating system update consists of updating the Reliant UNIX operating system software, i.e. the software on the "Reliant UNIX" CD-ROM. The operating system is for the most part packaged in PKG format.

The update, or to be more exact the `/etc/update` command, is controlled using options. An overview of the options available can be found using the `/etc/update –h` command (see also **OPTIONS**).

The operating system software is the same as is used in a new installation of a system. That is to say, there are no special update packages or update products. This ensures that both the new customer and the update customer have a system that is functionally identical.

There is always only one "Reliant UNIX" CD-ROM for an operating system release, and this can be used to update your system from various earlier operating system releases to the new operating system release. The update procedure can only update your system from an old to a new release, and not vice versa.

Only software on the "Reliant UNIX" CD-ROM can be updated. This means that both the operating system and the optional products installed later from the "Reliant UNIX" CD-ROM by the customer are updated. Software that is not on the "Reliant UNIX" CD-ROM cannot be updated as otherwise, for example, an automatic update would not be possible. Included in this is the software from add-on products that the customer installed from separate CD-ROMs (`COMM-SW-CD-ROM-MI`, `CD-ROM-SYS-MI`, `TV-SW-CD-ROM-MI` etc.). This software is, however, detected and processed if it contains dependencies relating to the operating system software, provided that these dependencies were made known to the system and thereby coded.

OPTIONS

`–c` (check) Turns on CHECK mode. This is the alternative to ARMED mode. In CHECK mode, the update does not change the system, which means that it can be called in this mode at any time. (Only the CHECK run log is stored on the system disk.) The update test procedures try to detect any problems that could prevent an update. If such problems are found, a warning is generally output that this problem can cause an error in ARMED mode and the update then continues. In some circumstances, a problem preventing an update also causes the cancellation of further tests in CHECK mode. This is the case if it is not possible to continue working properly (example: the installation medium is not available).

The steps of the update are only executed "virtually". This means, for example, that the package installation (`pkgadd`) tries to install a package in such a way that situations that might prevent an update are detected. The following are among the items checked:

- the system run level
- package dependencies
- disk space and number of inodes
- conflicts with previously installed packages
- incompatibilities with installed add-on products
- the consistency of the kernel components (the kernel is not, however, activated)

The history remains unchanged in CHECK mode.

−a (armed) Turns on ARMED mode. This is the alternative to CHECK mode. In ARMED mode, the update changes the operating system software, i.e. it updates it.

The steps of the update are "really" executed.

The read and write mode of the history is changed in ARMED mode.

The update test procedures are also executed in ARMED mode to detect problems that might prevent an update. If such problems are found, an error is generally returned and the update aborted. (If an error occurs, the abort can be avoided by using the −e option.)

−o (OTIP = Only The Installed Packages) In the standard case (i.e. without the −o option) all packages in the basic operating system are updated or installed. This is also the recommended procedure. There is, however, also the option of only installing packages that are already on the target system by specifying the −o option. This means that no package is installed for the first time, only existing packages are updated.

If, for example, a customer deleted the SIacc package due to a lack of hardware to optimize the disk space available, the /etc/update −a −o ... command installs all the packages in the basic operating system with the exception of the SIacc package.

Note:

1. Inconsistencies between the packages caused by arbitrary deletions are retained since the update cannot recognize why the packages were deleted. For example, there is no point in deleting a SIcdin package and leaving the SIcdinM language-dependent part on the system.
2. The −o option affects the basic operating system packages only. The rule for optional products installed by the customer (by means of sysadm using the "software_prod" − "products_on_CD" menu items) from the "Reliant UNIX" CD-ROM that they are only installed if they are already on the system still applies. So, for example, the RAID software is only ever installed (updated) if it was already on the target system before the update.
3. Some operating system updates require that specific packages be installed for compatibility reasons, even if they were not on the target system before the update; i.e. the −o option has no effect on these packages. The system informs you of this:

   Example:

```
------------------------------------------------------------------------
---
+++ INFO:    Scanning root disk for installed packages ..
+++ INFO:    You have chosen OTIP. The following packages may need
+++          to be installed for consistency reasons even
+++          though they may not already exist:
dwb, SIecnpw, SIerrlog, SIerrlogM, SIlar, SImabi, SImb2dbg, SIsysY, SIt
c20
------------------------------------------------------------------------
---
```

−e (errors will be skipped) In the standard case, (i.e. without the −e option), the update aborts in ARMED mode if an error preventing an update occurs. This is useful and does not cause any problems since the reentry ability of the update means that following error recovery the update resumes where it was aborted and continues. By selecting the −e option you can, however, ignore this and continue the update if errors occur. This is useful in some exceptional cases. The −e option should, however, be used with care; if errors are ignored, they can be the root cause of other problems. The result of an update using the −e option cannot be predicted if errors occur. For that reason, only experienced system administrators who can also interpret the side effects should use the −e option.

Note:

1. There are a few error situations that result in an error terminating the update despite the −e

option being set. This happens if the update cannot be continued properly. This is the case, for example, when:

- the installation medium is not available. The packages could therefore not be installed "virtually";
- the operating system version found on the system disk is not a suitable basis for the update installation: the update database contains no information as to how the "old" operating system version could be migrated to the "new" operating system version.

2. The `-e` option is only useful in ARMED mode. If an error that could prevent an update is detected in CHECK mode, a warning is output and the update continued.

`-i` (ignore history) Deletes the history of the update procedure.

The `/etc/update` command keeps a history of the update procedure currently being executed thus making it reentrant, i.□e. an aborted update procedure can be continued. Under normal circumstances (i.□e. without `-i`), an existing history is used and only the outstanding actions are performed. However, there are cases where the customer does not want to continue an aborted update procedure, rather would prefer to begin from the beginning again. This is possible by using the `-i` option to indicate that the history should be ignored.

Warning: The `-i` option is only intended for use by experts. You should avoid using it unless you absolutely have to. You must be aware of the consequences. Depending on the progress of the preceding update procedure, (whose history is now to be ignored) it is likely that some procedures will fail and manual intervention will be needed.

`-r` (read all parameters from file) In addition to the command line interface, the update has a call interface via a control file. Both interfaces are mutually exclusive: you either specify the required parameters in the command line, or you use a control file which contains the required parameters. The control file can have any name and the `-r` option must be specified as a parameter. If, for example, you created a control file with the name `config.update` in the `/home/julia` directory, you call the update using:

```
# /etc/update -r /home/julia/config.update
```

Note that no other command line parameters are permitted once the `-r` option is selected.

There is an example of such a control file containing the synopsis and additional notes (`/etc/update.cfg`) in the `SInus` package.

It is easier to prepare a completely automatic update (e.g. via a batch job) using a control file.

`-h` (help) Displays the command synopsis.

THE LOG

The update informs you both in CHECK mode and in ARMED mode about the individual actions that have taken place. The messages that are displayed on screen are also written to a file, which means that you always have an overview of the completed actions even when you are not present at the time. The log contains the messages from all calls of the same update, from both CHECK runs and ARMED runs. This simplifies considerably the tracing of the actions in terms of when they occurred. You must note, however, that the last update run is located at the end of the file.

The individual actions are separated visually from each other to improve clarity, since this log extends over several pages. Individual update calls are delimited by corresponding messages ("run of UPDATE started at [date]" or "run of UPDATE ended at [date]"). Within the update calls, the actions are defined more precisely by keywords:

`INFO` You are informed of "normal" operations. You are given this information on the current processing status on the active system. No actions are necessary.

`WARNING`
You are informed of unusual operations. This information has different impacts depending on the update mode:

1. CHECK mode □

There is a problem that needs closer inspection. This problem, if not resolved, can lead to an error (ERROR), i.e. usually abortion of the update, in ARMED mode. *Action*: Resolve this error. (Remark: you can see from the context how best to proceed. You can ignore the WARNING if this is indicated explicitly.)

Note that one problem can be the origin of other problems. If is therefore a good idea when a number of WARNINGs are returned to resolve the first one and then repeat the update in CHECK mode.

2. ARMED mode □
   There is a problem that needs closer inspection. This problem will not prevent an update but it should be resolved as soon as possible. *Action*: Resolve the problem as soon as possible.

ERROR

Informs you of operations that prevent updates. The program normally terminates (i.e. if the −e option is not set).

1. CHECK mode □
   There is a problem that also results in the program being terminated in CHECK mode since there is no point in continuing. This is, for example, necessary if a large part of the test cannot be executed at all because the installation medium is invalid. *Action*: Resolve the problem and restart the update.
2. ARMED mode □
   There is a problem that results in the program being terminated since there is no point in continuing. *Action*: Resolve the problem and restart the update.

Please note that the statement about ERRORs and WARNINGs can always only refer to the current update run, since the update itself cannot decide if a WARNING that occurred earlier is still valid and must be counted. A special case arising from this is when the same update attempt has to be called repeatedly (reentry); here, only the WARNINGs (which do not abort the update in ARMED mode) that occurred during the current run are counted. However, since the log contains all calls of the update attempt, you can also reference the earlier WARNINGs by paging back through the log. The update will advise you of this.

PROCEDURE

The update works through a series of actions, informing you at all times of the current state of progress. These actions are performed in both CHECK mode and ARMED mode. The CHECK run provides a detailed overview of the actions to be expected during the "actual" update.

Update of 5.43 → 5.44: Since you are moving here from a 32-bit to a 64-bit operating system, the system is restarted automatically in ARMED mode during the update. You do not need to intervene in any way.

AUTOMATIC OPERATION

General

The update was developed with "unattended operation" in mind. That means that no interaction with the user is required once it starts. The update either ends with successful execution or an appropriate return code if an error occurs. This gives the user the option of being able to execute the update within batch runs or other automated program runs.

It makes sense, for example, to execute the update when the system is not busy to avoid interaction with other processes. A practical approach is, for example: you had installed the SInus package, executed an update in CHECK mode and received information from the update that your system is ready for the planned update. You carried out a backup and would like to perform the update on Friday evening. You can ensure that the installation medium is mounted for batch execution, the default file systems are mounted, and the system is in the recommended single user mode. (Your batch file can naturally also take care of this.)

Example of a batch file (pseudocode):

```
/etc/update -r /etc/update.my_config.check
/* my control file for CHECK mode */
```

```
IF [ update is successful in CHECK mode ]
   THEN /etc/update -r /etc/update.my_config.armed
       /* my control file for ARMED mode */
        IF [ update is successful in ARMED ]
           THEN [ reboot the system ]
           ELSE [ wait for the system administrator ]
        FI
   ELSE [ wait for the system administrator since CHECK
          was unsuccessful ]
FI
```

Update return codes

The /etc/update program called delivers a return code once it is finished. This return code can be analyzed to establish the success/failure of this update process and initiate appropriate actions automatically.

The /etc/update command delivers the following return codes:

0 The /etc/update command was completed successfully. The current update operation was successful: No ERROR (ARMED mode) or ERROR and WARNING (CHECK mode) occurred.

1 The current update operation was not successful. The /etc/update command returns an error: An ERROR (ARMED mode), and an ERROR or a WARNING (CHECK mode) occurred.

2 The current update operation was not successful. The /etc/update command was called incorrectly.

3 The current update operation was not successful. The /etc/update command was interrupted (SIGINT).

99 The /etc/update command returns an error: An internal error occurred.

THE rcU INTERFACE TO EXTERNAL PROGRAMS

The update offers the option of including external procedures (programs). This functions in a similar way to the rc2 scripts of the operating system. For that reason, these external procedures are also called rcU (run commands Update). They must be stored in a defined directory and contain the following interface conventions. These external procedures are executed by the update in ASCII sequence.

The external procedures permit among other things:

- execution of customer-specific actions,
- processing of applications from third-party vendors,
- intervention in the update even after delivery of the CD-ROMs.

The external procedures are supplied with the necessary information by the update and must, for their part, report the success/failure of their activity to the update using their return code.

The SInus package contains an example of such a procedure which contains the interface definitions and further notes (/var/sadm/install/update/rcU/U00template).

PROCEDURE (COOKBOOK)

You received the "Reliant UNIX" CD-ROM containing a newer version of the operating system than that which is running on your system.

Preliminary tests

It makes sense to check the system as soon as possible (weeks before the actual update) to allow the early detection of problems.

1) Installing the update tool

   To do this, insert the "Reliant UNIX" CD-ROM in a CD-ROM drive that you can access from the system you want to update. Install the SInus package using sysadm via the "software_prod" menu item.

2) Mounting (mount)

   Mount the installation medium. (In this example locally in the CD-ROM drive):

```
# mount -F hs /dev/ios0/sdisk00xs0 /SInus
```

x = SCSI-ID (RM600 = 6, RM600-E = 0, RM400 = 5)

3) Calling (`update`)

Call the update in CHECK mode:

```
# /etc/update -c
```

The update now executes a range of tests. The entire operation is logged in the `/var/sadm/pkg/.prot.update` file. The individual test steps are identified by keywords.

The update lasts approx. 30 – 60 minutes in CHECK mode. Once finished, it outputs a report containing the following:

- if and which problems (`ERROR`s and `WARNING`s) occurred and how they can be resolved,
- which software was installed,
- which other actions were executed.

You now have an initial impression of whether your system is ready for an update. Extract the exact error descriptions from the `/var/sadm/pkg/.prot.update` test log. Solutions are suggested where possible. Repeat the update in CHECK mode once the errors have been resolved. Only when the update has been completed successfully in CHECK mode does it make sense to perform an update in ARMED mode (although always possible).

If incompatibilities with add-on products or software from an external source are reported, contact the regional office responsible or the third-party software supplier. This is particularly important if you are updating from 5.43 to 5.44. You should therefore pay special attention to the section below entitled "UPDATE AND ADD-ON PRODUCTS".

Final tests

Immediately before the update in ARMED mode, you should execute another test run. The report on the update in CHECK mode is naturally only valid if no system-relevant changes were made to the system in the time up to the update in ARMED mode. Since this could not be ruled out in the previous weeks, repeat Step 3). If possible, execute the test under the same conditions as you want to execute the update in ARMED mode.

No problems should generally occur now.

The update

Once the update has been completed successfully in CHECK mode, execute the update in ARMED mode.

0) Data backup

You performed a data backup.

1) Switch to single user mode.

This is strongly recommended. (The update can also be started in multiuser mode; this mode is only advisable for experienced system administrators who can resolve errors themselves should they occur.)

```
# init s
```

2) Starting the network

If you want to mount the installation medium using `nfs`, you must start the network via which you can reach the installation medium. Example: network on the `lce`:

```
# sh /etc/rc2.d/S69inet start
```

3) Mounting (`mount`)

Mount the installation medium. (In this example locally in the CD-ROM drive):

```
# mount -F hs /dev/ios0/sdisk00xs0 /SInus
```

x = SCSI-ID (RM600 = 6, RM600-E = 0, RM400 = 5)

4) Calling (`update`)

Call an update in ARMED mode:

```
# /etc/update -a
```

The system is now updated. You are informed of the progress.

The updates lasts approx. 0.5 – 3 hours. The duration depends on the type of update, the speed of the installation medium, the volume of software to be updated, the system configuration etc.

What to do in the event of errors

If the procedure is disrupted because of errors, you will be offered a possible solution. Resolve these errors. Because the update is re-entrant, you can restart it without having to work your way again through actions that have already been completed. The update knows the history and is continued from the point where it was interrupted. You should start the update as follows:

- If the problem arose when updating to 5.43 (i.e. update without automatic reboot), you should invoke the update using the same command again, e.g.:

  ```
  # /etc/update -a
  ```

- If the problem arose when updating from 5.43 to 5.44 (i.e. update with automatic reboot), the update informs you how to initiate re-entrant mode:
  - If the abort happened before the automatic reboot (i.e. 5.43 is still active), you should invoke the update using the same command again, e.g.:

    ```
    # /etc/update -a
    ```

  - If the abort happened after the automatic reboot (i.e. 5.44 is already active), boot the system in multiuser mode because it is now in single user mode as a result of the error:

    ```
    # init 2
    ```

    An `rc2` script now resumes control again.

- If the problem arose when updating from 5.44 (i.e. update without automatic reboot), you should invoke the update using the same command again, e.g.:

  ```
  # /etc/update -a
  ```

The update is continued from the point where it was interrupted.

You could also start the update again in CHECK mode. The update in CHECK mode also knows the history and also only starts from the point where the error occurred.

Caution:

Please remember that re-entrant mode is dependent on the history being read. If you introduce inconsistencies before you restart the update (e.g. manually delete packages already processed), the update will not be able to recognize this. It assumes in this case that these packages have already been updated; the resulting inconsistencies can cause errors.

You have to continue an interrupted update as quickly as possible in ARMED mode. Do not leave the system in a "half updated" state for a long period of time. In the case of updates where there is no automatic reboot, (5.43 → 5.44), you should not reboot if an error occurs before the update has finished as otherwise unforeseen problems may arise when "older" software and previously updated software are being used together.

Many commands under Reliant UNIX store temporary files in `/tmp` that you require. Please make sure that a "private" `cron` job does not clean up the directory at update runtime.

The update can only be started once at the same time.

Examples

Mounting various installation media

Note: The following abbreviations are used:

`Sys1C` = startup system with CD-ROM
`Sys1D` = startup system with disk

`Sys2` = system to be updated

Mount the installation medium located on/in:

Example a): CD-ROM located locally in the CD-ROM drive of the machine to be updated:

```
# mount -F hs /dev/ios0/sdisk00xs0 /SInus
```

Example b): CD-ROM in the CD-ROM drive in a machine accessed using `nfs`:

```
[Sys1C] # mount -F hs /dev/ios0/sdisk00xs0 /cdrom
[Sys1C] # share -oro,root=Sys2 /cdrom
[Sys2]  # #Start the network since the system is in single user mode
[Sys2]  # mount -F nfs Sys1C:/cdrom /SInus
```

Example c): Disk with a local copy of the installation CD-ROM:

```
# mount -F ufs /dev/ios0/sdisk00xs0 /SInus
```

Example d): Disk with a copy of the installation CD-ROM in a machine accessed using `nfs`:

```
[Sys1D] # mount -F ufs /dev/ios0/sdisk00xs0 /mnt
[Sys1D] # share -oro,root=Sys2 /mnt
[Sys2]  # #Start the network since the system is in single user mode
[Sys2]  # mount -F nfs Sys1D:/mnt /SInus
```

Starting the network when the installation media are mounted by nfs

The network is normally "down" in single user mode. It is therefore necessary to start this network to establish the connection between the machine with the installation medium `[Sys1C]` and the machine to be installed `[Sys2]` via `nfs`. The procedure for starting the network depends on the interface used:

Example a): Network on the `lce` (LAN Controller Ethernet):

```
# System [Sys2] is in single user mode:
sh /etc/rc2.d/S69inet start
# [Sys1C] should now be accessible, i.e.
ping [Sys1C]
# or
showmount -e [Sys1C]
# should work.
# Then mount installation medium:
mount -F nfs [Sys1C]:/cdrom /SInus
```

Example b): Network on the `lct` (LAN Controller Token Ring) and
Example c): Network on the `lcf` (LAN Controller FDDI):

```
# System [Sys2] is in single user mode
PATH=$PATH:/opt/bin
# stop the network
# (some old releases do not do this with init s)
sh /etc/rc2.d/S68lct stop # (or S68lcf stop for FDDI)
# CMX must be started
sh /etc/rc2.d/CMX start
sh /etc/rc2.d/S68lct start # (or S68lcf start for FDDI)
sh /etc/S69inet start
# [Sys1C] should now be accessible, i.e.
ping [Sys1C]
# or
showmount -e [Sys1C]
# should work.
# Then mount installation medium:
mount -F nfs [Sys1C]:/cdrom /SInus
```

Transferring configuration files

The configuration of the operating system is defined, among other things, by the configuration files. These system configuration files are retained when the update is installed. Therefore, you do not need to save these files separately before the update installation begins.

The configuration files include, for example:

```
/etc/conf/cf.d/mtune
/etc/conf/cf.d/stune
/etc/cron.d/queuedefs
/etc/default/login
/etc/inet/services
/etc/mail/mailsurr
/etc/profile
/etc/shells
/var/spool/cron/crontabs/root
```

Some configuration files may have changed when upgrading from one operating system version to another. The changes are entered in your configuration files automatically during the update installation. Each file that has to be changed is first saved by creating a copy of the file. The names of the copied files are created by adding a suffix to the existing file name. This ending contains the name of the future operating system version (so that the backup files can also be assigned uniquely at a later time). For example:

```
'/.profile' will be saved as '/.profile.pre_SINIX-Y5.43B0040'
```

These backup files are also retained following the update installation, but can be deleted manually if required at a later time.

The update now tries to incorporate the necessary changes in the existing configuration files. If this succeeds, the modified configuration files will be used in the future. These configuration files will not be overwritten when the operating system packages are loaded. While you will be informed of this, you do not need to do anything here. For example:

```
Trying to update your '/.profile' automatically. □
/.profile was modified
```

If, for whatever reasons, the necessary changes cannot be incorporated in the existing configuration files (e.□g. because the differences are too great), standard configuration files are copied to the system disk – as with a new installation. In this case you will receive the appropriate warning. For example:

```
+++ WARNING:  Unable to modify the '/.profile' configuration file automatically.
+++           '/.profile' is deleted to ensure that you get the new '/.profile'.
+++           The file currently active is backed up as '/.profile.pre_SINIX-Y5.4
3B0040'.
+++           We will use the newly installed file temporarily.
+++           Please combine the newly installed
+++           /.profile                and your
+++           /.profile.pre_SINIX-Y5.43B0040 into a
+++           /.profile                later.
```

This warning will be noted in the log and you will also receive a corresponding electronic mail. (Please note that the statement about WARNINGs can always only refer to the current update run. If reentries are necessary, the configuration files will of course only be updated once; such WARNINGs are then noted earlier in the log.)

The update installation is continued.

If the update installation has been completed successfully (without ERRORs), you will be referred to any problems that may have arisen. For example:

```
+++ INFO:    Report for 'update -a' :
+++ INFO:    We detected the following during the UPDATE runtime:
+++          Number of WARNINGs :       5
+++          Number of ERRORs   :       0
+++ INFO:    Excellent! The current UPDATE installation run ended successfully.
+++          (Please look at the WARNINGS in the log file named
+++          /var/sadm/install/../pkg/.prot.update)
+++
+++          Please reboot your system as soon as possible.
+++          Make sure that you have read the entire log file
```

```
+++          and that you are quite familiar with the results reported by
+++          the UPDATE installation.
```

You can look at the WARNINGs in the log and should resolve them as soon as possible.

The configuration files that the update installation was not able to update automatically, should now be updated manually (i.e. before the reboot). This is necessary because some configuration files are only used in the boot phase of the system. In the case of a manual update, you should proceed as described in the example below:

Change to the directory containing the relevant configuration file, in this case `/.profile`. Here you will find

- the file `.profile.pre_SINIX-Y5.43B0040`

  (= the `.profile` file you saved *prior* to the update. This is the configured file that was valid in the "old" operating system version.)
- the file `.profile`

  (= the new, unconfigured file. This file is valid in the "new" operating system version.)

Load both files into an editor. Then search in the `.profile.pre_SINIX-Y5.43B0040` file for the entries you made yourself (e.g. variable definitions etc.) as well as the entries made by add-on products (these can generally be identified by comments). Add these entries to the `.profile` file. And that's it.

Please note the following:

Under no circumstances should you reactivate the "old" configuration files, neither by loading them from the backup tape, nor by renaming the `*.pre_SINIX-*` files to their "old" file names. The "new" operating system version requires the files in their "new" format.

UPDATE AND ADD-ON PRODUCTS

As already mentioned, the update can only install software located on the installation medium, though dependencies on add-on products are considered. This is particularly relevant for the 5.43 → 5.44 update: Add-on products that can no longer run under 5.44 may place the success of the update in question. A multi-stage concept therefore exists for handling add-on products:

- You should carefully read the CHECK run log even if the report does not indicate warnings. The CHECK run log informs you of packages and associated products that can no longer be used following the update. The update handles incompatible products in accordance with the type of incompatibility:
  - Incompatible products are reported
  - Products containing kernel components but which do not influence the basic system if they are not in the kernel are deactivated.
  - Products that impair the operability of the system, or cause other serious problems because of their incompatibility, are deleted.
- Please contact your local Siemens office or the supplier for the respective add-on product in good time to get a more recent (compatible) version of the particular product. You can then install this new version when the operating system has been updated.

TYPICAL ERROR SITUATIONS AND THEIR SOLUTIONS

There are some typical error situations, the cause and resolution of which we will show below using examples.

Problem:

```
------------------------------------------------------------------------------
+++ INFO:    Rebuilding kernel ...
 The UNIX Operating System will now be rebuilt.
 This will take some time. Please wait.
/opt/conf/pack.d/nb/space.c  18: [error]:  2324 Undefined: 'NBMAXSESS'
/opt/conf/pack.d/nb/space.c  18: [error]:  2112 Constant expression expected
/opt/conf/pack.d/nb/space.c  18: [error]:  2032 Negative dimension not allowed
/opt/conf/pack.d/nb/space.c  19: [error]:  2324 Undefined: 'NBMAXNAMES'
/opt/conf/pack.d/nb/space.c  19: [error]:  2112 Constant expression expected
```

```
/opt/conf/pack.d/nb/space.c  19: [error]:  2032 Negative dimension not allowed
/opt/conf/pack.d/nb/space.c  20: [error]:  2142 Illegal initializer
/opt/conf/pack.d/nb/space.c  21: [error]:  2142 Illegal initializer
     2086 c1: errors: 8, warnings: 3
[error]:        9025 /usr/ic/bin/cc : frontend error while compiling file \
                                      '/opt/conf/pack.d/nb/space.c'
ERROR: '/opt/conf/pack.d/nb/space.c' will not compile properly
+++ ERROR:   There is a problem in your kernel configuration files.
+++          It is not possible to build a new kernel. Please
+++          check your kernel configuration files.
+++          See idbuild(1M) for details.
+++ INFO:    We will now copy a new standard kernel onto your system
+++          to permit a system reboot.
------------------------------------------------------------------------------
```

Cause:

The `/etc/conf/cf.d/mtune` file could not be updated automatically. Therefore all parameters that were entered in the `mtune` file by the add-on products are undefined. The newly loaded `mtune` file is on the system as well as the `mtune.pre_SINIX-Y5.43B0040` file, which is the `mtune` system file backed up before the update.

Solution:

Copy the parameters that are undefined in the `mtune` file from the `mtune.pre_SINIX-Y5.43B0040` file. In concrete terms:

```
# cd /etc/conf/cf.d/
# grep NBMAX mtune.pre_SINIX-Y5.43B0040 >> mtune
```

Ready. Restart the update which then continues on from the unsuccessful "Rebuilding kernel" stage.

Problem:

```
------------------------------------------------------------------------------
+++ INFO:    87  of      87  packages still have to be updated.
+++          Loading skeleton from /SInus/sinix/5_43/T1/skeleton ...
+++ WARNING: Loading of skeleton from /SInus/sinix/5_43/T1/skeleton could fail:
Processing Package DC/OSx (skeleton) from </SInus/sinix/5_43/T1/skeleton>.
    /etc/conf <no longer a directory>
    /tmp <no longer a directory>
    /usr/spool <no longer a directory>
    unable to stat filesystem mounted on </cdrom>
Space checking failed.
Installation of DC/OSx (skeleton) was suspended (interaction required).
-- processed 1 of 1 packages (100 %) --
+++          This may cause a fatal error in armed mode !
------------------------------------------------------------------------------
```

Cause:

The system is inconsistent: A file system (in this case `/cdrom`) is referred to as "mounted" (i.e. has an entry in the `mnttab`), but is not available.

The same problem occurs if, for example, the `df` command is called. This problem typically occurs if file systems are mounted using `nfs` but the connection no longer exists.

Solution:

To make the system consistent: unmount the relevant file system.

FILES

`/var/sadm/pkg/.prot.update`
: protocol of update installation

System Administrator's Reference Manual

updmemcpu(8)
RM400 only

NAME

`updmemcpu` - read and write CPU and DIMM information

SYNOPSIS

`/usr/bin/lar/updmemcpu [ –w ] [ –R ] [ –p` *cpuno* `] [ –c` *card* `] [ –b` *bank* `] [ –d` *dimm* `] [ –s` *state* `] [ –h ]`

DESCRIPTION

`updmemcpu` is required to read or write LAR status information for the CPU and storage DIMMs.

This information is stored in the NVMEM as well as in the HW_CONFIG database. It is required so that

a) the firmware can exclude faulty components when the system starts up (this is why the information is stored in the NVMEM).

b) the current configuration and status of the CPU and DIMM components can be displayed when the system is running.

The two information sources, i.e. the NVMEM and HW_CONFIG, are synchronized when the system boots and when serious errors occur.

If a status is modified or written (`–w` option), the values 0 (for "OK") and 1 (for "defect") are permitted (`–s` option).

If `updmemcpu` is used as the read program (no `–w` option) for the status, the values 0 (for "OK"), 1 (for "defect"), 2 (for "component not available") and 3 (error indicator) are returned.

OPTIONS

No argument specified:
Updates HW_CONFIG to the latest status with the information from the NVMEM.

`–w` Changes the contents of the NVMEM; the status is specified with the `–c`, `–p`, `–b`, `–d` or `–s` options.

`–w` option not specified:

The status of the specified component is returned. Components are specified with the `–c`, `–p`, `–b` or `–d` options.

`–p` *cpuno*
Specifies a CPU with the number *cpuno*. Permitted entries are in the range 0–3.

`–c` *card*
Specifies a memory card with the number *card*. Permitted entries are 1 or 2.

`–b` *bank*
Specifies a memory bank with the number *bank*. Permitted entries are in the range 1–4.

`–d` *dimm*
Specifies a DIMM with the number *dimm*. Permitted entries are in the range 1–8.

`–s` *state*
Specifies a status *state* to be set. Permitted entries are 0 ("OK") or 1 ("defect").

`–R` Sets LAR mode to "No defects and no LAR mode".

`–h` Outputs the usage.

EXAMPLES

`/usr/bin/lar/updmemcpu`
Updates the HW_CONFIG database to the latest status with the information from the NVMEM.

`/usr/bin/lar/updmemcpu -p 3`
Returns the status of the CPU with the number 3.

```
/usr/bin/lar/updmemcpu -w -c 1 -b 2 -d 3 -s 1
```

Sets the status in the NVMEM to "defect" for the DIMM with the number 2.3 on the first memory card.

SEE ALSO

lar(7),olr(7).

System Administrator's Reference Manual

vdadmin(8)

NAME

vdadmin - administration of virtual disks (vdisks) using Vdisk Lite (VL)

SYNOPSIS

vdadmin –X

vdadmin –c group –g *groupid* –n *statesave_vdisk* [–o *grouptype*] –L *conf_level*

vdadmin –c import –g *groupid* –d *disk_dev* –n *vdisk*

vdadmin –c select –g *groupid* –d *lun_dev1*,*lun_dev2* –n *vdisk*

vdadmin –c export –d *vdisk*

vdadmin –c mirror –d *vdisk1*,*vdisk2*

vdadmin –c base2select –d *vdisk1*,*vdisk2*

vdadmin –c stripe –d *vdisk1*,*vdisk2*[,...] –s *stripesize* –n *vdisk*

vdadmin –c concat –d *vdisk1*,*vdisk2*[,...] –n *vdisk*

vdadmin –c olr –x *iosname*

vdadmin –c olr –x *vdisk*,*piece_number*,*device_fs_name*

vdadmin –r –d *vdisk*

vdadmin –p create –d *volume* –n *vdisk1*[:*size1*][,...]

vdadmin –p delete –d *volume*

vdadmin –p split –d *pvdisk* –n *vdisk1*[:*size1*][,...]

vdadmin –p join –d *volume*

vdadmin –l [–v] [–d *vdisk1*[,*vdisk2* ...] | –g *groupid*]

vdadmin –q next [–g *groupid*]

vdadmin –q free {–g *groupid* | –d *vdisk*}

vdadmin –q imported | avail

vdadmin –R set | reset –d *vdisk1*[,*vdisk2* ...]

vdadmin –C "*comment*" –d *vdisk1*[,*vdisk2* ...]

vdadmin –N *host1*[,*host2* ...] –g *groupid*

vdadmin –u *oldnodename*,*newnodename*

vdadmin –B *oldserialno*,*newserialno*

vdadmin –t {start | unlock}

vdadmin –t end –k *transaction_key*

vdadmin –z node | cluster –h *node_name*

vdadmin –h *host1*[,*host2* ...]

The –h option cannot be called on its own, rather with each of the options {–c, –C, –r, –p, –q, –l, –R, –N}, in order to specify the host or hosts for which the operation is to apply.

vdadmin –S –k *transaction_key*

The –S –k *transaction_key* options can be used in most vdadmin calls that makes changes to the database, i.e. –c, –r, –p, –R, –C and –N. See the section further below on **Special options for cluster operation**.

DESCRIPTION

vdadmin is a command for creating and changing Vdisk Lite configurations [see vdisklite(7)]. The vdadmin database comprises the files vdisk.daps and vdiskgrp.daps in /etc/daps/daps. vdadmin

allows you to create mirror and/or stripe configurations.

The sequence of configuration steps with `vdadmin` is strictly defined:

- A new vdisk group must first be created (`-c group` option).
- Physical disks are then imported into this group (`-c import` option). This operation results in the creation of mirror or base vdisks.
- These (also termed low-level) vdisks can then be combined to form stripe vdisks (`-c stripe` option).
- The resulting vdisks can now be processed further to form concat vdisks through logical concatenation (`-c concat` option).
- The resulting vdisks can be partitioned into pieces of the desired size, called p-vdisks, (`-p create` option).
- A partitioned vdisk is called a volume.

While creation of stripe or concat vdisks and partitioning is possible, it is not mandatory. However, the sequence of these operations cannot be modified.

`vdadmin` is also required for OLR because changes must also be made in the DAPS (Data Pool Service) in this case. This call should only be initiated, however, from the user interface.

OPTIONS

`-B` *oldserialno,newserialno*

This option is only used in conjunction with the "blockmove" function for EMC Symmetrix raid boxes. Blockmove shifts data to other LUNs, which have different serial numbers. The `-B` option updates these serial numbers in the VL configuration database.

Attention:

The `-B` option should only be used by an experienced system administrator otherwise there is a risk that all data will be lost.

`-C` "*comment*"

Sets the comment field for the specified vdisk (`-d` option) with the text "*comment*" in the DAPS.

`-c` *keyword*

Makes changes in the DAPS.

The following entries are possible for *keyword*:

`group`
Create group. See the `-o` option below.

`import`
Import one physical disk into a group (result: base vdisk) or two or three physical disks into a group (result: mirror vdisk). This allows a physical disk to be used initially for VL.

`export`
The inverse operation to import: A base or mirror vdisk is resolved, the physical disks can then no longer be used for VL.

`mirror`
Two or three base vdisks are joined to form a mirror vdisk, the name of the first base vdisk is preserved (but the type changes), the second (and – if specified – the third) vdisk are lost.

`stripe`
Several (up to 99) base vdisks or several (up to 99) mirror vdisks are joined to form a stripe vdisk.

`concat`
Several (up to 99) base vdisks or several (up to 99) mirror vdisks are joined to form a concat vdisk.

`olr` Option only intended for the interface.

`select`
A vdisk of type `select` is created. Two physical devices must be imported.

Notes:

The type `select` is used in SRDF configurations on EMC Symmetrix.

The imported devices should always be the R1 and R2 site (master and slave site) of a LUN within an SRDF configuration.

The type `select` is like a mirror disk where only one piece can ever have the status "online".

`base2select`
A vdisk of type `select` is generated from two base vdisks.

Note:

The vdisks used should always be base vdisks on top of the R1 and R2 site of a LUN within an SRDF configuration.

`-d` *dev1*[,*dev2*,...]
Specifies the device (or devices) to be used as the source device for this operation. For example, `vdadmin -c concat -d vdisk1,vdisk2 -n vdisk3` causes `vdisk1` and `vdisk2` to be joined logically. The resulting vdisk is `vdisk3`.

Note:

A device in the case of the option is

`-c import`
a disk in the `ios` name space (e.g. to be specified in the form [`/dev/`]`ios0/sdisk035`);

`-c export`
a low-level vdisk;

`-c mirror`
a base vdisk or a mirror vdisk with two pieces;

`-p` a vdisk;

`-l` a vdisk;

`-c select`
the R1 and R2 site of an SRDF LUN;

`-c base2select`
a base vdisk on top of the R1 and R2 site of an SRDF LUN.

`-g` *groupid*
Specifies a group ID of a group of vdisks. Only possible for creating groups, importing vdisks and for queries.

`-h` *host1*[,*host2* ...]
Specifies one (or more) hosts for which the specified operation is to apply (`-c`, `-r`, `-p`, `-L`, `-R` or `-C` options).

`-L` *conf_level*
If `-L 0` is specified when creating a group (other entries not possible at present), this means that this group will be configured at system startup in `/etc/brc` by calling `dkconfig -vac`. This entry is only permitted for `-o local`.

`-l` Output information about all vdisks in a group.

When used together with the `-v` option, the pieces of the vdisks are also displayed. If the `-g` option is not specified, the vdisks in all groups are displayed. Note that only VL vdisks are displayed and not those flagged in `/etc/dktab`.

`-n` *vdisk1*[`:`*size1*][,...]
Specifies the resulting vdisk(s) from an operation. Example: The call `vdadmin -p create -d vdisk25 -n vdisk30:200mb,vdisk31:800mb,vdisk32,vdisk33` creates the vdisks specified with `-n` as p-vdisks on the `vdisk25` volume, where `vdisk30` is 200 Mbytes in size and `vdisk31` is 800 Mbytes in size and the remainder of `vdisk25` is divided equally between `vdisk32` and `vdisk33`.

−o *grouptype*
Only possible in conjunction with −c group.

grouptype defines the purpose of all vdisks of the group being created. The following entries are possible for *grouptype*:

local
To be specified if this group is only to be used from one computer. This is the default and can be omitted.

observe
To be specified if this group is to be used alternately by several computers.

rms To be specified if this group is to be controlled by RMS.

ops To be specified if the vdisks are for OPS or SPS.

opsres
To be specified if the mirror disks in this group are to support Oracle resilvering.

If observe, rms, ops or opsres is specified as the *grouptype*, the −h option must be used to specify a list of computers for which this group is to apply.

−p *keyword*
Partitions a vdisk into several smaller p-vdisks or edits the partitioning.

The following entries are possible for *keyword*:

create
Creates partitioning (see example under −n option).

delete
Deletes partitioning of a volume. All p-vdisks, located above the specified volume, are deleted.

split
A p-vdisk is subsequently split into several smaller p-vdisks, the syntax is the same as −p create.

join Two neighboring p-vdisks of a volume are combined to form a p-vdisk, the name of the second p-vdisk is lost.

−q *keyword*
Queries to the vdisk DAPS database.

The following entries are possible for *keyword*:

next Outputs the next free vdisk name (if a group is specified, the search commences from the number of the statesave vdisk in this group).

free Queries whether the specified vdisk name is already assigned.

imported
Outputs all imported physical disks.

avail
Queries which physical disks are not used and can be imported into VL.

−R set | reset
set sets the read-only attribute for the specified (currently not configured) vdisk.

reset resets the attribute again; this vdisk can be written to again following the next configuration.

−r The vdisk specified with the −d option (of type concat, stripe or statesave only!) is deleted.

−s *stripesize*
When a stripe vdisk is created, the stripe cluster size is set to the specified value. Specified in 512-byte blocks, the following values can be selected: 32, 64, 128 and 256.

−v (verbose) Only useful in combination with the −l option (refer to this option). If −v is entered, the pieces are also displayed for every vdisk shown.

−x Only for OLR operation, see −c olr and the comment above prior to the **OPTIONS** section.

–X Extended display of options.

Special options for cluster operation

If vdadmin is used in a cluster, the database (vdisk.daps and vdiskgrp.daps) is automatically copied to all nodes in the cluster. A cluster-wide lock file is created to ensure that different vdadmin calls on different nodes in the cluster do not impede one another. As soon as a vdadmin call is active, vdadmin cannot be called on any other node in the cluster.

In order to start a number of vdadmin calls, for example, from a shell script, a transaction can be initiated followed by several vdadmin calls in succession that only modify the database locally before the transaction is ended. The database is copied in the process to all nodes in the cluster.

–t start
: Starts a transaction. A lock file is created for the entire cluster, which protects the database throughout the cluster. Assuming the lock file could be created, the vdadmin –t start call creates a high number that must be used as a key for vdadmin calls.

–t end –k *transaction_key*
: Ends a previously started transaction. The currently valid transaction key *transaction_key* must always be provided if the –k option is specified.

–S –k *transaction_key*
: Prevents the database being distributed in the cluster. A transaction key *transaction_key* must always be provided with –k if the –S option is specified.

Example:

```
key=`vdadmin -tstart`
vdadmin -cgroup -g1 -nvdisk100 -S -k $key
vdadmin -cimport -g1 -n vdisk80 -d ios0/sdisk080,ios0/sdisk090 -S -k $key
vdadmin -cimport -g1 -n vdisk81 -d ios0/sdisk081,ios0/sdisk091 -S -k $key
vdadmin -t end -k $key
```

–t unlock
: If an error occurs in a transaction (e.g. a shell script crashes) or if the valid transaction key cannot be remembered, –t unlock can be used to cancel the lock file throughout the cluster.

This operation should only be performed if you are quite sure that a vdadmin session is not active on any node in the cluster!

–z node | cluster –h *node_name*
: This option is invoked automatically if a node is removed from a cluster. vdadmin -z cluster *node_name* is invoked on the removed node and vdadmin -z node *node_name* is invoked on a node that remains in the cluster. As a result, all vdisks on the node to be removed are deleted provided they are not available as local vdisks on this node. The *node_name* node is removed on the remaining cluster node from all non-local groups and, in addition, all local vdisks are removed from the *node_name* node.

This option should not be invoked manually.

–N *host1*[,*host2* ...] –g *groupid*
: This option modifies the list of host names for which this group is to apply.

Example: Assume you have an existing cluster (system1 and system2) and a vdisk group that applies for both systems (group 27). An additional node is now to be added to the cluster and this node is also to be able to access vdisks in group 27. The correct call is as follows:

```
vdadmin –N system1,system2,system3 -g 27
```

DIAGNOSTICS

When vdadmin is ended normally (successfully), the value 0 is returned. Otherwise, the return value is as follows:

1 (EX_ADDPIECE_IMPOSSIBLE)

One additional piece may be added to a base or mirror vdisk with two pieces.

2 (EX_BAD_O_OPTION)
The only entries permitted for the –o option are local, observe, ops, opsres or rms.

3 (EX_NOT_ON_TYPE)
An attempt was made to start an operation for an incorrect vdisk type, e.g. exporting a concat vdisk.

4 (EX_BAD_L_OPTION)
In the case of the –L option, only the null character (0) is currently permitted as a parameter.

5 (EX_NO_DISKVOLUME_DAPS)
The disk volume DAPS cannot be opened.

6 (EX_DISK_UNAVAIL)
A physical disk could not be imported to Vdisk Lite, because it was already being used for another purpose.

7 (EX_BAD_C_CMD)
Only one of the following keywords can be specified for the –c option: group, import, export, mirror, concat, stripe, olr.

8 (EX_PIECE_UNAVAIL)
A piece of a concat, stripe or mirror vdisk is not available.

9 (EX_TOO_FEW_SOURCES)
Too few source vdisks were specified with the –d option.

10 (EX_TOO_MANY_SOURCES)
Too many source vdisks were specified with the –d option.

11 (EX_TOO_MANY_TARGETS)
Too few target vdisks were specified with the –n option.

12 (EX_UNEQUAL_SIZE)
The pieces of a stripe or mirror vdisk are different sizes.

13 (EX_BAD_OPTION)
vdadmin was called with an incorrect option or the entries for the –x option were incorrect.

14 (EX_BAD_P_CMD)
The only keywords that can be used for the –p option are create, join, split or delete.

15 (EX_BAD_COLON_SIZE)
Incorrect size (with "*size*" for the –n option) specified when partitioning or splitting a p-vdisk.

16 (EX_STATESAVE_BUSY)
The statesave vdisk in the group is still configured. No changes can be made to the configuration.

17 (EX_BAD_GROUP)
No number was specified for the –g option.

19 (EX_NO_GID)
The –g option was not specified when creating a group or importing a disk.

20 (EX_NO_VDDAPS)
The vdisk DAPS could not be opened.

21 (EX_DAPS_ERROR)
An internal DAPS error has occurred (not specified in any greater detail).

22 (EX_GROUP_NOEXIST)
An attempt was made to import a disk into a non-existent group or it was established with –q –g *groupid* that this group does not exist.

23 (EX_VDISK_EXISTS)
An attempt was made to create a vdisk that already exists or it was established with –q –d *vdiskx* that *vdiskx* already exists.

24 (EX_DISK_IMPORTED)

This disk is already imported.

25 (EX_DKPART)
: The disk could not be opened for changing the partition table.

26 (EX_DKIOCGETPART)
: The partition table on the disk could not be read.

27 (EX_DKIOCSETPART)
: The partition table on the disk could not be written.

28 (EX_VDISK_NOEXIST)
: A specified target vdisk does not exist.

29 (EX_UNEQUAL_GROUPS)
: Only vdisks from the same group can be joined to form mirror, stripe or concat vdisks.

34 (EX_INSERT_FAIL)
: Something could not be inserted in the DAPS.

35 (EX_TOO_FEW_TARGETS)
: No or too few target vdisk(s) were specified with the `-n` option.

37 (EX_NO_SYN_GEO)
: An attempt was made to use a disk for `Vdisk Lite` that does not use synthetic geometry (import or OLR).

38 (EX_BAD_STRIPESIZE)
: Only the values 32, 64, 128 and 256 (specified in 512-byte blocks) are valid as the *stripesize*.

39 (EX_NO_SPAN)
: When creating a partition or splitting a p-vdisk, the partitioning specified did not cover the complete volume (or the complete p-vdisk).

40 (EX_VDCONF_OPEN)
: The vdisk configuration device could not be opened. A `dkconfig` call is probably active at present.

41 (EX_VDISK_CONFIGURED)
: An action was attempted for a configured vdisk for which the vdisk cannot be configured.

42 (EX_PART_IMPOSSIBLE)
: Something went wrong when setting up the partitioning of a volume or a p-vdisk.

43 (EX_TARGET_MULTI)
: A vdisk was specified twice with the `-n` option.

44 (EX_SOURCE_MULTI)
: A device was specified twice with the `-d` option.

45 (EX_P_VDISK_REMOVE)
: P-vdisks cannot be removed with the `-r` option (only with `-p delete` for the underlying volume).

46 (EX_VDISK_UNAVAIL)
: The requested operation cannot be executed for this vdisk because it is not available.

47 (EX_BAD_QUERY)
: The only keywords permitted with the `-q` option are `free`, `next`, `imported` or `avail`.

49 (EX_JOIN_IMPOSSIBLE)
: It is not possible to join two p-vdisks because they are located on different volumes or the second p-vdisk does not start where the first one finishes.

50 (EX_GROUP_EXISTS)
: An existing group ID was used in the `vdadmin -c -g` *groupid* or `vdadmin -q -g` *groupid* call.

51 (EX_NO_HOSTLIST)
: When using `-o observe`, `-o ops`, `-o opsres` or `-o rms`, a list of systems must be specified with the `-h` option.

54 (EX_TARGET_IN_DKTAB)

A vdisk was specified with the `-n` option which already appears in the `/etc/dktab` file.

55 (`EX_OLR_RECONF`)
Online reconfiguration failed.

56 (`EX_DISK_OPEN_FAILED`)
The new disk could not be opened during an OLR operation.

57 (`EX_DISK_TOO_SMALL`)
It was discovered during an OLR operation that the new disk is too small.

58 (`EX_UNAME`)
A `utsname()` call failed.

59 (`EX_NO_VDISKGRP_DAPS`)
The `/etc/daps/daps/vdiskgrp.daps` file could not be opened.

60 (`EX_NO_PATH_DAPS`)
The disk path DAPS must be opened for a `vdadmin -c olr -x` *iosname* call. This did not work.

61 (`EX_PATH_NOT_FOUND`)
The sought entry was not found in the disk path DAPS.

62 (`EX_BAD_R_CMD`)
Only `set` or `reset` can be entered for the `-R` option.

63 (`EX_WHOOPS`)
Return value for various internal errors that should never occur.

64 (`EX_UNEQUAL_TYPES`)
The pieces of a concat vdisk must all be of the same type.

65 (`EX_NO_CLOS_CMD`)
The commands required for cluster operation could not be found.

66 (`EX_NO_TRANSACTION_KEY`)
A transaction key must be submitted with the `-k` option when a transaction is ended (`-t end`).

67 (`EX_BAD_T_OPTION`)
Only one of the parameters {`start` |`end` | `unlock`} can be specified with the `-t` option.

68 (`EX_COD_LOCK_FAIL`)
The `vdadmin -t start` call failed. The lock file is probably enabled by another `vdadmin` session.

69 (`EX_COD_SEND_FAIL`)
The database could not be sent. This error should never occur.

70 (`EX_COD_UNLOCK_FAIL`)
The resetting of the log file failed. This error should never occur.

71 (`EX_NO_K_OPTION`)
If the `-S` option is specified, the `-k` option must also be specified.

72 (`EX_DAPS_LOCKED`)
The lock file is enabled and the database cannot be modified.

73 (`EX_BAD_TRANSACTION_KEY`)
An incorrect transaction key was specified.

74 (`EX_BAD_N_CMD`)
If the `-N` option is specified, the `-g` option must also be specified.

75 (`EX_NO_NODE_DAPS`)
The `node.daps` file that must be present in a cluster could not be accessed.

76 (`EX_NODE_UNKNOWN`)
An attempt was made to add an unknown node to a vdisk group.

EXAMPLES

A complex example now follows.

The hardware requirements are as follows:

```
ios0/scon01 53C770 chan:2
  ios0/sport01 Id: 8 SE 16 Bit 10MHz
    ios0/sdisk010 MP48 -- MP48
    ios0/sdisk011 MP48 -- MP48
    ios0/sdisk012 MP48 -- MP48
    ios0/sdisk013 MP48 -- MP48
    ios0/sdisk014 MP48 -- MP48
    ios0/sdisk015 MP48 -- MP48
    ios0/sdisk016 MP48 -- MP48
ios0/scon02 53C770 chan:1
  ios0/sport02 Id: 8 SE 16 Bit 10MHz
    ios0/sdisk020 MP48 -- MP48
    ios0/sdisk021 MP48 -- MP48
    ios0/sdisk022 MP48 -- MP48
    ios0/sdisk023 MP48 -- MP48
    ios0/sdisk024 MP48 -- MP48
    ios0/sdisk025 MP48 -- MP48
    ios0/sdisk026 MP48 -- MP48
```

The following should be configured:

Two groups of vdisks, i.e.

- Group 1 for database devices

  Memory capacity required:

  - 1 x 8-Gbyte RAID Level 1+0 (mirror and stripe) => 2 * 2 physical disks
  - 2 x 1-Gbyte RAID Level 1+0 with at least 4-Gbyte RAID Level 1+0 in reserve => 2 * 2 physical disks

  In total 2 * 4 disks.

- Group 2 for file systems

  Memory capacity required:

  - 1 x 5-Gbyte (RAID Level 1+0) with an option to increase to 8 Gbytes => 2 * 2 physical disks
  - 1 x 2-Gbyte unmirrored and unstriped, remainder in reserve for unmirrored and unstriped => 2 physical disks

All physical disks are thus planned for.

The configuration steps are as follows:

- Create group 1 where `vdisk0` is created as a statesave device for group 1. Configuration is performed automatically at every startup:

  ```
  > vdadmin -c group -g 1 -n vdisk0 -L 0
  ```

- The following command uses the principle of "mirror first, then stripe" to create an optimal vdisk configuration:

  ```
  > vdadmin -c import -g 1 -n vdisk1 -d ios0/sdisk010,ios0/sdisk020
  > vdadmin -c import -g 1 -n vdisk2 -d ios0/sdisk011,ios0/sdisk021
  > vdadmin -c import -g 1 -n vdisk3 -d ios0/sdisk012,ios0/sdisk022
  > vdadmin -c import -g 1 -n vdisk4 -d ios0/sdisk013,ios0/sdisk023
  ```

- A large stripe volume is now created:

  ```
  > vdadmin -c stripe -n vdisk10 -d vdisk1,vdisk2,vdisk3,vdisk4
  ```

- The stripe vdisk created is partitioned into the required p-vdisks:

  ```
  > vdadmin -p create -d vdisk10 \
  -n disk11:8GB,vdisk12:1GB,vdisk13:1GB,vdisk14
  ```

The reserve `vdisk14` disk is now assigned the remainder of the volume, and is thus some 6 Gbytes in size (if

one assumes that a disk of type MP48 is 4 Gbytes large).

All database devices have now been created.

When this group is first configured, the stripe vdisk (`vdisk10`) cannot be configured because a piece of the underlying mirror disks must first be set to the status "online". This is what you must do here:

(configure statesave and mirror vdisks)

`dkconfig -vc vdisk0 vdisk1 vdisk2 vdisk3 vdisk4`

`dkmirror -o0 -e1 vdisk1` (start catchup)

`dkmirror -o0 -e1 vdisk2` (start catchup)

`dkmirror -o0 -e1 vdisk3` (start catchup)

`dkmirror -o0 -e1 vdisk4` (start catchup)

(configure the remaining vdisks in this group)

`dkconfig -vc vdisk10 vdisk11 vdisk12 vdisk13 vdisk14`

If this group is to be configured subsequently, the following call will suffice:

`dkconfig -vcg1`

Now to the file system devices.

- Create group 2, whereby `vdisk100` is created as a statesave device for group 2:

  ```
  > vdadmin -c group -g 2 -n vdisk100
  ```

- The following command uses the principle of "mirror first, then stripe" to create an optimal vdisk configuration:

  ```
  > vdadmin -c import -g 2 -n vdisk101 -d ios0/sdisk014,ios0/sdisk024
  > vdadmin -c import -g 2 -n vdisk102 -d ios0/sdisk015,ios0/sdisk025
  ```

- An 8-GByte volume is now created with RAID Level 1+0 (`vdisk110`),

  ```
  > vdadmin -c stripe -n vdisk110 -d vdisk101,vdisk102
  ```

- A 5-Gbyte (`vdisk111`) partition is created on this volume for the file system. The remainder of the volume is kept in reserve (`vdisk112`).

  ```
  > vdadmin -p create -d vdisk110 -n vdisk111:5GB,vdisk112
  ```

- Because not everything is to be mirrored, the remaining disks are imported in an unmirrored state:

  ```
  > vdadmin -c import -g2 -d ios0/sdisk016 -n vdisk103
  > vdadmin -c import -g2 -d ios0/sdisk026 -n vdisk104
  ```

- `vdisk103` is partitioned for the second file system:

  ```
  > vdadmin -p create -d vdisk103 -n vdisk113:2GB,vdisk114
  ```

The `vdisk112`, `vdisk114` and `vdisk104` disks are therefore still available in reserve in this group.

As with group 1, the status of the mirror disks must be set following their initial configuration, so that the overlying stripe vdisk can be configured, e.g.:

`dkconfig -vcg2` (ignore error messages here)

`dkmirror -o0 -e1 vdisk101` (start catchup)

`dkmirror -o0 -e1 vdisk102` (start catchup)

`dkconfig -vcg2`

FILES

`/etc/daps/daps/vdisk.daps`

`/etc/daps/daps/vdiskgrp.daps`

SEE ALSO

`vdisklite(7)` .

System Administrator's Reference Manual

wsinit(8)
RM400 only

NAME

`wsinit -` make console devices and init graphics console

SYNOPSIS

`wsinit [-f]`

DESCRIPTION

`wsinit` is run early in the system startup via `/etc/inittab`. Its main purpose is to build the graphics' console STREAMS multiplexer, but it is also used to relink `/dev/console` et al. to its proper device.

When started, `wsinit` checks the console byte and the teleservice flags in NVRAM to see what kind of console the system should use. It then removes `/dev/console`, `/dev/syscon`, `/dev/systty` and `/dev/contty`, and relinks them with the proper device (either `/dev/consgraf`, `/dev/constty`, `/dev/ttyc1` (teleservice) or `/dev/null`).

If the console byte indicates a graphics console, `wsinit` proceeds to build the STREAMS multiplexer. Therefore, the `/etc/default/workstations` file is read to find out which driver is to be linked below the multiplexer, and how the device files for the upper part of the multiplexer are to be named.

Then, all entries in `/dev` that have the multiplexer's major number are removed, and new device entries are created according to `/etc/default/workstations`. The new devices are linked to the driver devices via the `I_PLINK ioctl` in order to build the multiplexer configuration.

OPTIONS

`-f` Force building of the STREAMS multiplexer even if the console byte in NVRAM says that we don't have a graphics console.

DIAGNOSTICS

The `wsinit` command will try to do its task even if some things fail. It will partially fail under the following conditions:

- If `/dev/nvram` cannot be found or read.
- If the console devices cannot be removed.
- If the console devices cannot be created.
- If the multiplexer devices cannot be removed.
- If the multiplexer devices cannot be created.
- If the multiplexer devices cannot be linked via `I_PLINK`.

NOTES

`wsinit` will fail partially if the root filesystem is mounted read only, which happens if it is dirty. However, the STREAMS multiplexer is built, and console I/O is possible. It is *not* possible to change the console device while the root filesystem is dirty.

FILES

`/dev/console`
`/dev/syscon`
`/dev/systty`
`/dev/contty`
`/dev/nvram`
`/etc/default/workstations`

/etc/inittab

SEE ALSO

display(7).