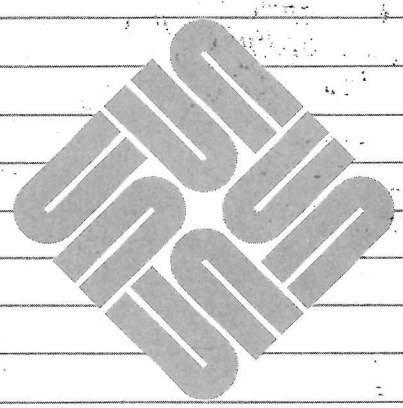
sun® microsystems

SunOS Reference Manual



Part Number: 800-3827-10
Revision A of 27 March, 1990

NAME

intro – introduction to system maintenance and operation commands

DESCRIPTION

This section contains information related to system bootstrapping, operation and maintenance. It describes all the server processes and daemons that run on the system, as well as standalone (PROM monitor) programs.

An 8V section number means one or more of the following:

- The man page documents System V behavior only.
- The man page documents default SunOS behavior, and System V behavior as it differs from the default behavior. These System V differences are presented under SYSTEM V section headers.
- The man page documents behavior compliant with *IEEE Std 1003.1-1988* (POSIX.1).

Disk formatting and labeling is done by **format**(8S). Bootstrapping of the system is described in **boot**(8S) and **init**(8). The standard set of commands run by the system when it boots is described in **rc**(8). Related commands include those that check the consistency of file systems, **fsck**(8); those that mount and unmount file systems, **mount**(8); add swap devices, **swapon**(8); force completion of outstanding file system I/O, **sync**(2); shutdown or reboot a running system **shutdown**(8), **halt**(8), and **reboot**(8); and, set the time on a machine from the time on another machine **rdate**(8C).

Creation of file systems is discussed in **mkfs**(8) and **newfs**(8). File system performance parameters can be adjusted with **tunefs**(8). File system backups and restores are described in **dump**(8) and **restore**(8).

Procedures for adding new users to a system are described in **adduser**(8), using **vipw**(8) to lock the password file during editing. **panic**(8S) which describes what happens when the system crashes, **savecore**(8) which can be used to analyze system crash dumps. Occasionally useful as adjuncts to the **fsck**(8) file system repair program are **clri**(8), **dcheck**(8), **icheck**(8), and **ncheck**(8).

Configuring a new version of the kernel requires using the program **config**(8); major system bootstraps often require the use of **mkproto**(8). New devices are added to the **/dev** directory (once device drivers are configured into the system) using **makedev**(8) and **mknod**(8). The **installboot**(8S) command can be used to install freshly compiled programs. The **catman**(8) command preformats the on-line manual pages.

Resource accounting is enabled by the **accton** command, and summarized by **sa**(8). Login time accounting is performed by **ac**(8). Disk quotas are managed using **quot**(8), **quotacheck**(8), **quotaon**(8), and **repquota**(8).

A number of servers and daemon processes are described in this section. The **update**(8) daemon forces delayed file system I/O to occur and **cron**(8) runs periodic events (such as removing temporary files from the disk periodically). The **syslogd**(8) daemon maintains the system error log. The **init**(8) process is the initial process created when the system boots. It manages the reboot process and creates the initial login prompts on the various system terminals, using **getty**(8). The Internet super-server **inetd**(8C) invokes all other internet servers as needed. These servers include the remote shell servers **rshd**(8C) and **rexecd**(8C), the remote login server **rlogind**(8C), the FTP and TELNET daemons **ftpd**(8C), and **telnetd**(8C), the TFTP daemon **tftpd**(8C), and the mail arrival notification daemon **comsat**(8C). Other network daemons include the 'load average/who is logged in' daemon **rwhod**(8C), the routing daemon **routed**(8C), and the mail daemon **sendmail**(8).

If network protocols are being debugged, then the protocol debugging trace program **trpt**(8C) is often useful. Remote magnetic tape access is provided by **rsh** and **rmt**(8C). Remote line printer access is provided by **lpd**(8), and control over the various print queues is provided by **lpc**(8). Printer cost-accounting is done through **pac**(8).

Network host tables may be gotten from the ARPA NIC using **gettable**(8C) and converted to UNIX-system-usable format using **htable**(8).

RPC and NFS daemons

RPC and NFS daemons include:

portmap	used by RPC based services.
ypbind	used by the Network Information Service (NIS) to locate the NIS server. Note: the Network Information Service (NIS) was formerly known as Sun Yellow Pages (YP). The functionality of the two remains the same; only the name has changed.
biod	used by NFS clients to read ahead to, and write behind from, network file systems.
nfsd	the NFS server process that responds to NFS requests on NFS server machines.
ypserv	the NIS server, typically run on each NFS server.
rstatd	the server counterpart of the remote speedometer tools.
mountd	the **mount** server that runs on NFS server machines and responds to requests by other machines to mount file systems.
rwalld	used for broadcasting messages over the network.

LIST OF MAINTENANCE COMMANDS

Name	Appears on Page	Description
ac	**ac**(8)	login accounting
acctcms	**acctcms**(8)	command summary from per-process accounting records
acctcon1	**acctcon**(8)	connect-time accounting
acctcon2	**acctcon**(8)	connect-time accounting
acctdisk	**acct**(8)	miscellaneous accounting commands
acctdusg	**acct**(8)	miscellaneous accounting commands
acctmerg	**acctmerg**(8)	merge or add total accounting files
accton	**acct**(8)	miscellaneous accounting commands
accton	**sa**(8)	system accounting
acctprc1	**acctprc**(8)	process accounting
acctprc2	**acctprc**(8)	process accounting
acctwtmp	**acct**(8)	miscellaneous accounting commands
adbgen	**adbgen**(8)	generate adb script
add_client	**add_client**(8)	create a diskless network bootable NFS client on a server
add_services	**add_services**(8)	provide software installation services for any architecture
adduser	**adduser**(8)	procedure for adding new users
adv	**adv**(8)	advertise a directory for remote access with RFS
analyze	**old-analyze**(8)	postmortem system crash analyzer
arp	**arp**(8C)	address resolution display and control
audit	**audit**(8)	audit trail maintenance
auditd	**auditd**(8)	audit daemon
audit_warn	**audit_warn**(8)	audit daemon warning script
automount	**automount**(8)	automatically mount NFS file systems
biod	**nfsd**(8)	NFS daemons
boot	**boot**(8S)	start the system kernel, or a standalone program
bootparamd	**bootparamd**(8)	boot parameter server
C2conv	**c2conv**(8)	convert system to or from C2 security
C2unconv	**c2conv**(8)	convert system to or from C2 security
captoinfo	**captoinfo**(8V)	convert a termcap description into a terminfo description
catman	**catman**(8)	create the cat files for the manual
change_login	**change_login**(8)	control screen blanking and choice of login utility
chargefee	**acctsh**(8)	shell procedures for accounting
check4	**set4**(8)	check the virtual address space limit flag in a module
chown	**chown**(8)	change owner
chroot	**chroot**(8)	change root directory for a command
chrtbl	**chrtbl**(8)	generate character classification table
ckpacct	**acctsh**(8)	shell procedures for accounting

client	**client**(8)	add or remove diskless Sun386i systems
clri	**clri**(8)	clear inode
colldef	**colldef**(8)	convert collation sequence source definition
comsat	**comsat**(8C)	biff server
config	**config**(8)	build system configuration files
copy_home	**copy_home**(8)	fetch default startup files for new home directories
crash	**crash**(8)	examine system images
cron	**cron**(8)	clock daemon
dbconfig	**dbconfig**(8)	initializes the dial box
dcheck	**dcheck**(8)	file system directory consistency check
devinfo	**devinfo**(8S)	print out system device information
devnm	**devnm**(8V)	device name
diskusg	**diskusg**(8)	generate disk accounting data by user
dkctl	**dkctl**(8)	control special disk operations
dkinfo	**dkinfo**(8)	report information about a disk's geometry and partitioning
dmesg	**dmesg**(8)	collect system diagnostic messages to form error log
dname	**dname**(8)	print RFS domain and network names
dodisk	**acctsh**(8)	shell procedures for accounting
dorfs	**dorfs**(8)	initialize, start and stop RFS automatically
dump	**dump**(8)	incremental file system dump
dumpfs	**dumpfs**(8)	dump file system information
edquota	**edquota**(8)	edit user quotas
eeprom	**eeprom**(8S)	EEPROM display and load utility
etherd	**etherd**(8C)	Ethernet statistics server
etherfind	**etherfind**(8C)	find packets on Ethernet
exportfs	**exportfs**(8)	export and unexport directories to NFS clients
extract_unbundled	**extract_unbundled**(8)	extract and execute unbundled-product installation scripts
fastboot	**fastboot**(8)	reboot/halt the system without checking the disks
fasthalt	**fastboot**(8)	reboot/halt the system without checking the disks
fingerd	**fingerd**(8C)	remote user information server
format	**format**(8S)	disk partitioning and maintenance utility
fpa_download	**fpa_download**(8)	download to the Floating Point Accelerator
fparel	**fparel**(8)	Sun FPA online reliability tests
fpaversion	**fpaversion**(8)	print FPA version, load microcode
fpurel	**fpurel**(8)	perform tests the Sun Floating Point Co-processor
fpuversion4	**fpuversion4**(8)	print the Sun-4 FPU version
fsck	**fsck**(8)	file system consistency check and interactive repair
fsirand	**fsirand**(8)	install random inode generation numbers
ftpd	**ftpd**(8C)	TCP/IP Internet File Transfer Protocol server
fumount	**fumount**(8)	force unmount of an advertised RFS resource
fusage	**fusage**(8)	RFS disk access profiler
fuser	**fuser**(8)	identify processes using a file or file structure
fwtmp	**fwtmp**(8)	manipulate connect accounting records
gettable	**gettable**(8C)	get DARPA Internet format host table from a host
getty	**getty**(8)	set terminal mode
gid_allocd	**uid_allocd**(8C)	UID and GID allocator daemons
gpconfig	**gpconfig**(8)	initialize the Graphics Processor
grpck	**grpck**(8V)	check group database entries
gxtest	**gxtest**(8S)	stand alone test for the Sun video graphics board
halt	**halt**(8)	stop the processor
hostrfs	**hostrfs**(8)	Convert IP addresses to RFS format
htable	**htable**(8)	convert DoD Internet format host table
icheck	**icheck**(8)	file system storage consistency check

idload	**idload**(8)	RFS user and group mapping
ifconfig	**ifconfig**(8C)	configure network interface parameters
imemtest	**imemtest**(8S)	stand alone memory test
in.comsat	**comsat**(8C)	biff server
inetd	**inetd**(8C)	Internet services daemon
in.fingerd	**fingerd**(8C)	remote user information server
infocmp	**infocmp**(8V)	compare or print out terminfo descriptions
in.ftpd	**ftpd**(8C)	TCP/IP Internet File Transfer Protocol server
init	**init**(8)	process control initialization
in.named	**named**(8C)	Internet domain name server
in.rexecd	**rexecd**(8C)	remote execution server
in.rlogind	**rlogind**(8C)	remote login server
in.routed	**routed**(8C)	network routing daemon
in.rshd	**rshd**(8C)	remote shell server
in.rwhod	**rwhod**(8C)	system status server
installboot	**installboot**(8S)	install bootblocks in a disk partition
install_small_kernel	**install_small_kernel**(8)	install a small, pre-configured kernel
installtxt	**installtxt**(8)	create a message archive
in.talkd	**talkd**(8C)	server for talk program
in.telnetd	**telnetd**(8C)	TCP/IP TELNET protocol server
in.tftpd	**tftpd**(8C)	TCP/IP Trivial File Transfer Protocol server
in.tnamed	**tnamed**(8C)	TCP/IP Trivial name server
intr	**intr**(8)	allow a command to be interruptible
iostat	**iostat**(8)	report I/O statistics
ipallocd	**ipallocd**(8C)	Ethernet-to-IP address allocator
kadb	**kadb**(8S)	adb-like kernel and standalone-program debugger
keyenvoy	**keyenvoy**(8C)	talk to keyserver
keyserv	**keyserv**(8C)	server for storing public and private keys
kgmon	**kgmon**(8)	generate a dump of the operating system's profile buffers
lastlogin	**acctsh**(8)	shell procedures for accounting
ldconfig	**ldconfig**(8)	link-editor configuration
link	**link**(8V)	exercise link and unlink system calls
listen	**nlsadmin**(8)	network listener service administration for RFS
lockd	**lockd**(8C)	network lock daemon
logintool	**logintool**(8)	graphic login interface
lpc	**lpc**(8)	line printer control program
lpd	**lpd**(8)	printer daemon
mailstats	**mailstats**(8)	print statistics collected by sendmail
makedbm	**makedbm**(8)	make a NIS ndbm file
MAKEDEV	**makedev**(8)	make system special files
makekey	**makekey**(8)	generate encryption key
mc68881version	**mc68881version**(8)	print the MC68881 mask number and approximate clock rate
mconnect	**mconnect**(8)	connect to SMTP mail server socket
mkfile	**mkfile**(8)	create a file
mkfs	**mkfs**(8)	construct a file system
mknod	**mknod**(8)	build special file
mkproto	**mkproto**(8)	construct a prototype file system
modload	**modload**(8)	load a module
modstat	**modstat**(8)	display status of loadable modules
modunload	**modunload**(8)	unload a module
monacct	**acctsh**(8)	shell procedures for accounting
monitor	**monitor**(8S)	system ROM monitor
mountd	**mountd**(8C)	NFS mount request server

mount	**mount**(8)	mount and unmount file systems
mount_tfs	**mount_tfs**(8)	mount and dismount TFS filesystems
named	**named**(8C)	Internet domain name server
ncheck	**ncheck**(8)	generate names from i-numbers
ndbootd	**ndbootd**(8C)	ND boot block server
netconfig	**netconfig**(8C)	PNP boot service
netstat	**netstat**(8C)	show network status
newaliases	**newaliases**(8)	rebuild the data base for the mail aliases file
newfs	**newfs**(8)	create a new file system
newkey	**newkey**(8)	create a new key in the publickey database
nfsd	**nfsd**(8)	NFS daemons
nfsstat	**nfsstat**(8C)	Network File System statistics
nlsadmin	**nlsadmin**(8)	network listener service administration for RFS
nslookup	**nslookup**(8C)	query domain name servers interactively
nsquery	**nsquery**(8)	RFS name server query
nulladm	**acctsh**(8)	shell procedures for accounting
old-analyze	**old-analyze**(8)	postmortem system crash analyzer
pac	**pac**(8)	printer/plotter accounting information
panic	**panic**(8S)	what happens when the system crashes
ping	**ping**(8C)	send ICMP ECHO_REQUEST packets to network hosts
pnpboot	**pnpboot**(8C)	pnp diskless boot service
pnpd	**pnpd**(8C)	PNP daemon
pnp.s386	**pnpboot**(8C)	pnp diskless boot service
portmap	**portmap**(8C)	TCP/IP port to RPC program number mapper
praudit	**praudit**(8)	print contents of an audit trail file
prctmp	**acctsh**(8)	shell procedures for accounting
prdaily	**acctsh**(8)	shell procedures for accounting
prtacct	**acctsh**(8)	shell procedures for accounting
pstat	**pstat**(8)	print system facts
pwck	**pwck**(8V)	check password database entries
pwdauthd	**pwdauthd**(8C)	server for authenticating passwords
quotacheck	**quotacheck**(8)	file system quota consistency checker
quotaoff	**quotaon**(8)	turn file system quotas on and off
quotaon	**quotaon**(8)	turn file system quotas on and off
quot	**quot**(8)	summarize file system ownership
rarpd	**rarpd**(8C)	TCP/IP Reverse Address Resolution Protocol server
rc	**rc**(8)	command scripts for auto-reboot and daemons
rc.boot	**rc**(8)	command scripts for auto-reboot and daemons
rc.local	**rc**(8)	command scripts for auto-reboot and daemons
rdate	**rdate**(8C)	set system date from a remote host
rdump	**dump**(8)	incremental file system dump
reboot	**reboot**(8)	restart the operating system
renice	**renice**(8)	alter nice value of running processes
repquota	**repquota**(8)	summarize quotas for a file system
restore	**restore**(8)	incremental file system restore
rexd	**rexd**(8C)	RPC-based remote execution server
rexecd	**rexecd**(8C)	remote execution server
rfadmin	**rfadmin**(8)	RFS domain administration
rfpasswd	**rfpasswd**(8)	change RFS host password
rfstart	**rfstart**(8)	start RFS
rfstop	**rfstop**(8)	stop the RFS environment
rfuadmin	**rfuadmin**(8)	RFS notification shell script
rfudaemon	**rfudaemon**(8)	Remote File Sharing daemon

rlogind	**rlogind**(8C)	remote login server
rmail	**rmail**(8C)	handle remote mail received via uucp
rm_client	**rm_client**(8)	remove an NFS client
rmntstat	**rmntstat**(8)	display RFS mounted resource information
rmt	**rmt**(8C)	remote magtape protocol module
route	**route**(8C)	manually manipulate the routing tables
routed	**routed**(8C)	network routing daemon
rpc.etherd	**etherd**(8C)	Ethernet statistics server
rpcinfo	**rpcinfo**(8C)	report RPC information
rpc.lockd	**lockd**(8C)	network lock daemon
rpc.mountd	**mountd**(8C)	NFS mount request server
rpc.rexd	**rexd**(8C)	RPC-based remote execution server
rpc.rquotad	**rquotad**(8C)	remote quota server
rpc.rstatd	**rstatd**(8C)	kernel statistics server
rpc.rusersd	**rusersd**(8C)	network username server
rpc.rwalld	**rwalld**(8C)	network rwall server
rpc.sprayd	**sprayd**(8C)	spray server
rpc.statd	**statd**(8C)	network status monitor
rpc.yppasswdd	**yppasswdd**(8C)	server for modifying NIS password file
rpc.ypupdated	**ypupdated**(8C)	server for changing NIS information
rquotad	**rquotad**(8C)	remote quota server
rrestore	**restore**(8)	incremental file system restore
rshd	**rshd**(8C)	remote shell server
rstatd	**rstatd**(8C)	kernel statistics server
runacct	**acctsh**(8)	shell procedures for accounting
runacct	**runacct**(8)	run daily accounting
rusage	**rusage**(8)	print resource usage for a command
rusersd	**rusersd**(8C)	network username server
rwalld	**rwalld**(8C)	network rwall server
rwhod	**rwhod**(8C)	system status server
sa	**sa**(8)	system accounting
savecore	**savecore**(8)	save a core dump of the operating system
sendmail	**sendmail**(8)	send mail over the internet
set4	**set4**(8)	set the virtual address space limit flag in a module
setsid	**setsid**(8V)	set process to session leader
showfhd	**showfhd**(8C)	showfh daemon run on the NFS servers
showfh	**showfh**(8C)	print full pathname of file from the NFS file handle
showmount	**showmount**(8)	show all remote mounts
shutacct	**acctsh**(8)	shell procedures for accounting
shutdown	**shutdown**(8)	close down the system at a given time
skyversion	**skyversion**(8)	print the SKYFFP board microcode version number
sprayd	**sprayd**(8C)	spray server
spray	**spray**(8C)	spray packets
start_applic	**start_applic**(8)	generic application startup procedures
startup	**acctsh**(8)	shell procedures for accounting
statd	**statd**(8C)	network status monitor
sticky	**sticky**(8)	mark files for special treatment
sundiag	**sundiag**(8)	system diagnostics
suninstall	**suninstall**(8)	install and upgrade the SunOS operating system
swapon	**swapon**(8)	specify additional device for paging and swapping
sys-config	**sys-config**(8)	configure a system or administer configuration information
syslogd	**syslogd**(8)	log system messages
sys-unconfig	**sys-unconfig**(8)	undo a system's configuration

talkd	**talkd**(8C)	server for talk program
telnetd	**telnetd**(8C)	TCP/IP TELNET protocol server
tfsd	**tfsd**(8)	TFS daemon
tftpd	**tftpd**(8C)	TCP/IP Trivial File Transfer Protocol server
tic	**tic**(8V)	terminfo compiler
tnamed	**tnamed**(8C)	TCP/IP Trivial name server
trpt	**trpt**(8C)	transliterate protocol trace
ttysoftcar	**ttysoftcar**(8)	enable/disable carrier detect
tunefs	**tunefs**(8)	tune up an existing file system
turnacct	**acctsh**(8)	shell procedures for accounting
tzsetup	**tzsetup**(8)	set up old-style time zone information in the kernel
uid_allocd	**uid_allocd**(8C)	UID and GID allocator daemons
umount	**mount**(8)	mount and unmount file systems
umount_tfs	**mount_tfs**(8)	mount and dismount TFS filesystems
unadv	**unadv**(8)	unadvertise a Remote File Sharing resource
unconfigure	**unconfigure**(8)	reset the network configuration for a Sun386i system
unlink	**link**(8V)	exercise link and unlink system calls
unset4	**set4**(8)	unset the virtual address space limit flag in a module
update	**update**(8)	periodically update the super block
user_agentd	**user_agentd**(8C)	user agent daemon
uucheck	**uucheck**(8C)	check the UUCP directories and Permissions file
uucico	**uucico**(8C)	file transport program for the UUCP system
uuclean	**uuclean**(8C)	uucp spool directory clean-up
uucleanup	**uucleanup**(8C)	UUCP spool directory clean-up
uucpd	**uucpd**(8C)	UUCP server
uusched	**uusched**(8C)	the scheduler for the UUCP file transport program
uuxqt	**uuxqt**(8C)	execute remote command requests
vipw	**vipw**(8)	edit the password file
vmstat	**vmstat**(8)	report virtual memory statistics
wtmpfix	**fwtmp**(8)	manipulate connect accounting records
ypbatchupd	**ypbatchupd**(8C)	NIS batch update daemon
ypbind	**ypserv**(8)	NIS server and binder processes
ypinit	**ypinit**(8)	build and install NIS database
ypmake	**ypmake**(8)	rebuild NIS database
yppasswdd	**yppasswdd**(8C)	server for modifying NIS password file
yppoll	**yppoll**(8)	version of NIS map at NIS server
yppush	**yppush**(8)	force propagation of changed NIS map
ypserv	**ypserv**(8)	NIS server and binder processes
ypset	**ypset**(8)	point ypbind at a particular server
ypsync	**ypsync**(8)	collect most up-to-date NIS maps
ypupdated	**ypupdated**(8C)	server for changing NIS information
ypxfr	**ypxfr**(8)	transfer NIS map from NIS server to here
zdump	**zdump**(8)	time zone dumper
zic	**zic**(8)	time zone compiler

NAME

ac – login accounting

SYNOPSIS

/usr/etc/ac [**–w** *wtmp*] [**–p**] [**–d**] [*username*] ...

DESCRIPTION

ac produces a printout giving connect time for each user who has logged in during the life of the current *wtmp* file. A total is also produced.

The accounting file **/var/adm/wtmp** is maintained by **init**(8) and **login**(1). Neither of these programs creates the file, so if it does not exist no connect-time accounting is done. To start accounting, it should be created with length 0. On the other hand if the file is left undisturbed it will grow without bound, so periodically any information desired should be collected and the file truncated.

OPTIONS

–w *wtmp*
: Specify an alternate *wtmp* file.

–p
: Print individual totals; without this option, only totals are printed.

–d
: Printout for each midnight to midnight period. Any *people* will limit the printout to only the specified login names. If no *wtmp* file is given, **/var/adm/wtmp** is used.

FILES

/var/adm/wtmp

SEE ALSO

login(1), **utmp**(5V), **init**(8), **sa**(8)

NAME

acctdisk, acctdusg, accton, acctwtmp – overview of accounting and miscellaneous accounting commands

SYNOPSIS

/usr/lib/acct/acctdisk

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

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

/usr/lib/acct/acctwtmp *reason*

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**(8) 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 **/etc/utmp**, as described in **utmp**(5V). The programs described in **acctcon**(8) convert this file into session and charging records, which are then summarized by **acctmerg**(8).

Process accounting is performed by the 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**(8) summarize this data for charging purposes; **acctcms**(8) 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 the format described in **acct**(5)) can be merged and summarized into total accounting records by **acctmerg** (see **tacct** format in **acct**(5)). **prtacct** (see **acctsh**(8)) 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.

accton without arguments turns process accounting off. If *filename* is given, it must be the name of an existing file, to which the kernel appends process accounting records (see **acct**(2V) and **acct**(5)). You must be super-user to use this command.

acctwtmp writes a **utmp**(5V) record to its standard output. The record contains the current time and a string of characters that describe the *reason*. The login name for this record is set to **@@acct** (see **utmp**(5V)). *reason* must be a string of 8 or fewer characters, numbers, $, or SPACE characters. If *reason* contains a SPACE character, it must be enclosed in double quotes. For example, the following are suggestions for use in reboot and shutdown procedures, respectively:

```
acctwtmp uname >> /var/adm/wtmp
acctwtmp fsave >> /var/adm/wtmp
```

OPTIONS

acctdusg

–u *filename*

Place records consisting of those file names for which **acctdusg** charges no one in *filename* (a potential source for finding users trying to avoid disk charges).

–p *filename*

Use *filename* as the password file, rather than **/etc/passwd**. (See **diskusg**(8) for more details.)

FILES

/etc/passwd	used for login name to user ID conversions
/usr/lib/acct	holds all accounting commands listed in section 8 of this manual
/var/adm/pacct	current process accounting file
/var/adm/wtmp	login/logoff history file

SEE ALSO

acctcom(1), acct(2V), acct(5), utmp(5V), acctcms(8), acctcon(8), acctmerg(8), acctprc(8), acctsh(8), diskusg(8), fwtmp(8), runacct(8)

NAME

acctcms – command summary from per-process accounting records

SYNOPSIS

/usr/lib/acct/acctcms [**–cjnst**] *filename* ...

/usr/lib/acct/acctcms [**–a** [**po**] [**cjnstpo**] *filename* ...

DESCRIPTION

acctcms reads one or more *filename*s, normally in the form described in **acct**(5). It adds all records for processes that executed identically-named commands, sorts them, and writes them to the 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 processes.

–s Any file names 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.

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.

EXAMPLES

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
```

SEE ALSO

acctcom(1), **acct**(2V), **acct**(5), **utmp**(5V), **acct**(8), **acctcon**(8), **acctmerg**(8), **acctprc**(8), **acctsh**(8), **fwtmp**(8), **runacct**(8)

BUGS

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.

NAME

acctcon1, acctcon2 – connect-time accounting

SYNOPSIS

/usr/lib/acct/acctcon1 [**–pt**] [**–l** *file*] [**–o** *file*]

/usr/lib/acct/acctcon2

DESCRIPTION

acctcon1

acctcon1 converts a sequence of login/logoff records read from its standard input to a sequence of records, one per login session. Its input should normally be redirected from **/var/adm/wtmp**. Its output is ASCII, giving device, user ID, login name, prime connect time (seconds), non-prime connect time (seconds), session starting time (numeric), and starting date and time.

acctcon2

acctcon2 expects as input a sequence of login session records and converts them into total accounting records (see **tacct** format in **acct**(5)).

OPTIONS

acctcon1

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

–t Test mode. **acctcon1** maintains a list of lines on which 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.

–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. Hang-up, 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**(8) and **utmp**(5V).

–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.

EXAMPLES

These commands are typically used as shown below. The file **ctmp** is created only for the use of **acctprc**(8) commands:

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

FILES

/var/adm/wtmp

SEE ALSO

acctcom(1), **login**(1), **acct**(2V), **acct**(5), **utmp**(5V), **acct**(8), **acctcms**(8), **acctmerg**(8), **acctprc**(8), **acctsh**(8), **fwtmp**(8), **init**(8), **runacct**(8)

BUGS

The line usage report is confused by date changes. Use **wtmpfix** (see **fwtmp**(8)) to correct this situation.

NAME

acctmerg – merge or add total accounting files

SYNOPSIS

/usr/lib/acct/acctmerg [**–aiptuv**] [*filename*...]

DESCRIPTION

acctmerg reads its standard input and up to nine additional files, all in the **tacct** format (see **acct**(5)) 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 of **tacct.**

–i Input files are in ASCII version of **tacct.**

–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 <filename1 >filename2
        edit file2 as desired ...
acctmerg –i <filename2 >filename1
```

SEE ALSO

acctcom(1), **acct**(2V), **acct**(5), **utmp**(5V), **acct**(8), **acctcms**(8), **acctcon**(8), **acctprc**(8), **acctsh**(8), **fwtmp**(8), **runacct**(8)

NAME

acctprc1, acctprc2 – process accounting

SYNOPSIS

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

/usr/lib/acct/acctprc2

DESCRIPTION

acctprc1

acctprc1 reads input in the form described by **acct**(5), adds login names corresponding to user IDs, then writes for each process an ASCII line giving user ID, login name, prime CPU time (ticks), non-prime CPU time (ticks), and mean memory size (in pages). If *ctmp* is given, it is expected to be the name of a file containing a list of login sessions, in the form described in **acctcon**(8), sorted by user ID and login name. If this file is not supplied, it obtains login names from the password file. The information in *ctmp* helps it distinguish among different login names that share the same user ID.

acctprc2

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

EXAMPLES

These commands are typically used as shown below:

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

FILES

/etc/passwd

SEE ALSO

acctcom(1), **acct**(2V), **acct**(5), **utmp**(5V), **acct**(8), **acctcms**(8), **acctcon**(8), **acctmerg**(8), **acctsh**(8), **cron**(8), **fwtmp**(8), **runacct**(8)

BUGS

Although it is possible to distinguish among login names that share user IDs for commands run from the command line, it is difficult to do this for those commands run by **cron**(8), for example. More precise conversion can be done by faking login sessions on the console using the **acctwtmp** program in **acct**(8).

NAME

chargefee, ckpacct, dodisk, lastlogin, monacct, nulladm, prctmp, prdaily, prtacct, runacct, shutacct, startup, turnacct – shell procedures for accounting

SYNOPSIS

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

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

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

/usr/lib/acct/lastlogin

/usr/lib/acct/monacct *number*

/usr/lib/acct/nulladm *filename*

/usr/lib/acct/prctmp *filename*

/usr/lib/acct/prdaily [**–cl**] [*mmdd*]

/usr/lib/acct/prtacct *filename* [*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

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 during the night.

ckpacct

ckpacct should be initiated by **cron**(8) every hour. It periodically checks the size of **/var/adm/pacct**. If the size exceeds *blocks*, 1000 by default, **turnacct** is called with the argument **switch**. If the number of free disk blocks in the **/usr** file system falls below 500, **ckpacct** automatically turns off the collection of process accounting records using the **off** argument to **turnacct**. When at least this number of blocks is restored, accounting is activated again. This feature is sensitive to the frequency at which **ckpacct** is executed, usually by **cron**.

dodisk

dodisk should be executed by **cron** to perform the disk accounting functions. By default, it does disk accounting on the 4.2 file systems in **/etc/fstab**. *filename*s specify the one or more filesystem names where disk accounting will be done. If *filename*s are used, disk accounting will be done on these filesystems only. They should be the special file names of mountable filesystems.

lastlogin

lastlogin is invoked by **runacct** to update **/var/adm/acct/sum/loginlog**, which shows the last date on which each person logged in. **lastlogin** deletes the entries of users no longer in **/etc/passwd** and creates new entries.

monacct

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 executed by **cron**(8) on the first day of each month. **monacct** creates summary files in **/var/adm/acct/fiscal** and restarts summary files in **/var/adm/acct/sum**.

nulladm

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

prctmp

prctmp can be used to print the session record file with headings (normally **/var/adm/acct/nite/ctmp** created by **acctcon1** (see **acctcon**(8)). The heading specifies device, user ID, login name, prime connect time (in seconds), non-prime connect time (in seconds), session starting time (numeric) and starting date and time.

prdaily

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. Previous daily reports are cleaned up and therefore inaccessible after each invocation of **monacct**.

prtacct

prtacct can be used to format and print any total accounting (**tacct**) file with headings. See Chapter 8 in the *System and Network Administration* manual, for an explanation of this output.

runacct

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**(8).

shutacct

shutacct should be invoked during a system shutdown (usually in **/etc/shutdown**) to turn process accounting off and append a "reason" record to **/var/adm/wtmp**. If *reason* is not specified, **shutdown** is provided as a default reason.

startup

startup should be called by **/etc/rc** to turn the accounting on whenever the system is brought up.

turnacct

turnacct is an interface to **accton** (see **acct**(8)) to turn process accounting **on** or **off**. The **switch** argument turns accounting off, 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 **cron** and used to keep **pacct** to a reasonable size. This command is restricted to the super-user.

OPTIONS

dodisk

–o Do a slower version of disk accounting by login directory. *filename*s should be mount points of mounted filesystem.

prdaily

–c Prints a report of exceptional resource usage by command. This may be used on current day's accounting data only.

–l Print a report of exceptional usage by login ID for the specifed date.

FILES

/etc/fstab	list of file systems **/var/adm/fee** accumulator for fees
/var/adm/pacct	current file for per-process accounting
/var/adm/pacct*	used if pacct gets large and during execution of daily accounting procedure
/var/adm/wtmp	login/logoff summary
/usr/lib/acct/ptelus.awk	limits for exceptional usage by login id
/usr/lib/acct/ptecms.awk	limits for exceptional usage by command name
/var/adm/acct/nite	working directory
/usr/lib/acct	directory of accounting commands
/var/adm/acct/sum	summary directory, should be saved

SEE ALSO

acctcom(1), **acct**(2V), **acct**(5), **utmp**(5V), **acct**(8), **acctcms**(8), **acctcon**(8), **acctmerg**(8), **acctprc**(8), **cron**(8), **diskusg**(8), **fwtmp**(8), **runacct**(8)

System and Network Administration

NAME

adbgen – generate adb script

SYNOPSIS

/usr/lib/adb/adbgen *filename*.**adb** ...

DESCRIPTION

adbgen makes it possible to write **adb**(1) scripts that do not contain hard-coded dependencies on structure member offsets. The input to **adbgen** is a file named *filename*.**adb** which contains **adbgen** header information, then a null line, then the name of a structure, and finally an **adb** script. **adbgen** only deals with one structure per file; all member names are assumed to be in this structure. The output of **adbgen** is an **adb** script in *filename*. **adbgen** operates by generating a C program which determines structure member offsets and sizes, which in turn generates the **adb** script.

The header lines, up to the null line, are copied verbatim into the generated C program. Typically these include C **#include** statements to include the header files containing the relevant structure declarations.

The **adb** script part may contain any valid **adb** commands (see **adb**(1)), and may also contain **adbgen** requests, each enclosed in {}s. Request types are:

- Print a structure member. The request form is {*member*,*format*}. *member* is a member name of the *structure* given earlier, and *format* is any valid **adb** format request. For example, to print the **p_pid** field of the *proc* structure as a decimal number, you would write {**p_pid,d**}.
- Reference a structure member. The request form is {*****member,*base*}. *member* is the member name whose value is desired, and *base* is an **adb** register name which contains the base address of the structure. For example, to get the **p_pid** field of the *proc* structure, you would get the proc structure address in an **adb** register, say <f, and write {***p_pid,<f**}.
- Tell **adbgen** that the offset is ok. The request form is {**OFFSETOK**}. This is useful after invoking another **adb** script which moves the **adb** *dot*.
- Get the size of the *structure*. The request form is {**SIZEOF**}. **adbgen** replaces this request with the size of the structure. This is useful in incrementing a pointer to step through an array of structures.
- Get the offset to the end of the structure. The request form is {**END**}. This is useful at the end of the structure to get **adb** to align the **dot** for printing the next structure member.

adbgen keeps track of the movement of the **adb** *dot* and emits **adb** code to move forward or backward as necessary before printing any structure member in a script. **adbgen**'s model of the behavior of **adb**'s **dot** is simple: it is assumed that the first line of the script is of the form *struct_address/adb text* and that subsequent lines are of the form *+/adb text*. This causes the *adb* **dot** to move in a sane fashion. **adbgen** does not check the script to ensure that these limitations are met. **adbgen** also checks the size of the structure member against the size of the **adb** format code and warns you if they are not equal.

EXAMPLE

If there were an include file **x.h** which contained:

```
struct x {
        char    *x_cp;
        char    x_c;
        int     x_i;
};
```

Then an **adbgen** file (call it **script.adb**) to print it would be:

```
#include "x.h"
x
./"x_cp"16t"x_c"8t"x_i"n{x_cp,X}{x_c,C}{x_i,D}
```

After running **adbgen** the output file **script** would contain:

```
16t"x_c"8t"x_i"nXC+D"" ./"x_cp"16t"x_c"8t"x_i"nXC+D
```

To invoke the script you would type:

```
x$<script
```

FILES

/usr/lib/adb/*	**adb** scripts for debugging the kernel

SEE ALSO

adb(1), **kadb**(8S)

Debugging Tools

BUGS

adb syntax is ugly; there should be a higher level interface for generating scripts.

Structure members which are bit fields cannot be handled because C will not give the address of a bit field. The address is needed to determine the offset.

DIAGNOSTICS

Warnings about structure member sizes not equal to **adb** format items and complaints about badly formatted requests. The C compiler complains if you reference a structure member that does not exist. It also complains about & before array names; these complaints may be ignored.

NAME

add_client – create a diskless network bootable NFS client on a server

SYNOPSIS

/usr/etc/install/add_client [**–inpv**] [**–a** *kernel-arch*] [**–e** *exec-path*] [**–f** *share-path*] [**–h** *home-path*] [**–k** *kvm-path*] [**–m** *mail-path*] [**–r** *root-path*] [**–s** *swap-path*] [**–t** *term-type*] [**–y** *yptype*] [**–z** *swapsize*] [*client* ...]

DESCRIPTION

add_client adds an NFS client to a server. It can only be run by the super-user.

A default standard layout is used to set up the client's environment, but most pathnames can be overridden with the appropriate option, or menu field change.

Before you can add a client, you must first make sure that the Internet and Ethernet addresses for *client* are listed in the Network Interface Service (NIS) hosts database (if the server is running the NIS service), or in the server's **/etc/hosts** and **/etc/ethers** databases, respectively. If **add_client** cannot find the client entry in the hosts database it aborts the operation. If there is no client entry in the **/etc/ethers** database, **add_client** issues a warning to update this file while adding the client.

The default root and swap partitions are **/export/root/***client* and **/export/swap/***client*, respectively.

add_client updates the **/etc/bootparams** file on the server but not the bootparams database in the NIS service (if used).

If the server is not running as an NIS master, **add_client** issues a warning to indicate that the database is out of date and the NIS master should be updated.

add_client updates the server's **/etc/exports** file to allow client's root access to each client's root file system. It also exports each client's swap file accordingly. Note: the system administrator should verify that the **/etc/exports** file contains correct information, and that file systems are exported to the correct users and groups. Refer to **exportfs**(8) for details on exporting file systems.

If the **–i** or **–p** option is not specified, at least one *client* argument must be supplied on the command line.

OPTIONS

–i Interactive. Bring up a full-screen menu interface to **add_client**.

–n Print the working parameters and exit without doing anything. This is used to verify what parameters **add_client** will use before actually doing anything.

–p Display a short version of all client information, If *client*s are specified on the command line, only display information for those clients. When combined with the –v option, a long version of client information is displayed.

–v Verbose. Report information about the client as steps are performed.

–a *kernel-arch* Specify the client kernel architecture (for instance, **sun3**, **sun4**, **sun4c**...). **add_client** prompts for the kernel architecture when unable to determine the correct value.

–e *exec-path* Set the pathname of the directory in which the executables for the architecture specified by **–a**. The client mounts **/export/exec/***arch.rel* as **/usr**. See WARNINGS.

–f *share-path* Set the pathname of the share directory, which is normally a link to **/usr/share**.

–h *home-path* Set the pathname of the directory for the client's home. The default is **/home/***server-name*.

–k *kvm-path* Set the pathname of the directory containing the client's kernel executables. See WARNINGS.

–m *mail-path* Set the pathname of the client's mail directory. The default is **/var/spool/mail**.

–r *root-path* Set the pathname of parent directory for client root directories; *root/client* is the pathname of the client's root directory. The default is **/export/root/***client-name*.

-s *swap-path* Set the pathname of parent directory for client swap files; *swap/client* is the pathname of the client's swap file. The default is **/export/swap/***client-name*.

-t *term-type* Set the terminal type of the client's console.

-y *yptype* Indicate the type of NIS server or if client is to be an NIS client; it can be **client** or **none**. The **none** argument results in the NIS service being disabled on the client. The default is **client**.

-z *swapsize* Reserve *swapsize* bytes for the client's swap file. **swapsize** can be flagged as kilobytes, blocks, or megabytes, with the **k**, **b**, or **m** suffixes, respectively. The default is 16Mb, and bytes are used when no units are specified.

FILES

/etc/bootparams
/etc/ethers
/etc/exports
/etc/hosts
/export/exec/proto.root.*release*
architecture independent base for the client root file system
/tftpboot.*client-ipaddr* link to **/tftpboot/boot.***arch*

SEE ALSO

add_services(8), **bootparamd(8)**, **exportfs**(8), **ndbootd**(8C), **rm_client**(8), **suninstall**(8)

Installing SunOS 4.1

DIAGNOSTICS

add_client: must be super-user
You must be root to use **add_client**.

NOTES

The Network Information Service (NIS) was formerly known as Sun Yellow Pages (YP). The functionality of the two remains the same; only the name has changed.

WARNINGS

The **-e** *exec-path* and the **-k** *kvm-path* options should not be used since the correct paths are determined when the adding the client's architecture service. See **add_services**(8).

NAME

add_services – provide software installation services for any architecture

SYNOPSIS

/usr/etc/install/add_services

DESCRIPTION

add_services is a menu-based program to setup a system as a server and/or to add additional software categories or other architecture releases. It is used to provide support to diskless clients, dataless clients, or just to act as a file server. **add_services** can only be run by the super-user.

add_services updates the **/etc/exports** file (see **exports**(5) and **exportfs**(8)) to export the necessary file systems to become a file server. After running **add_services**, the system administrator should verify this file to make sure that the new services have been exported to the correct groups.

FILES

/etc/hosts	hosts database, host must be in this database or in the Network Interface Service (NIS) hosts map
/etc/exports	database of exported file systems, service related directories must be exported
/tftpboot	add_services sets up this directory in order to provide boot service to clients

SEE ALSO

exports(5), **add_client**(8), **exportfs**(8), **rm_client**(8), **suninstall**(8)

Installing SunOS 4.1

NOTES

The Network Information Service (NIS) was formerly known as Sun Yellow Pages (YP). The functionality of the two remains the same; only the name has changed.

NAME

adduser – procedure for adding new users

DESCRIPTION

To add an account for a new user, the system administrator (or super-user):

- Create an entry for the new user in the system password files.
- Create a home directory for the user, and change ownership so the new user owns that directory.
- Optionally set up skeletal dot files for the new user (**.cshrc**, **.login**, **.profile**...).
- If the account is on a system running the Network Interface Service (NIS), take additional measures.

USAGE

Making an Entry in the Password File

To add an entry for the new login name on a local host, first edit the **/etc/passwd** file — inserting a line for the new user. This must be done with the password file locked, for instance, by using **vipw**(8), and the insertion must be made above the line containing the string:

+::0:0:::

This line indicates that additional accounts can be found in the NIS service.

To add an entry for the new login name into the NIS service, add an identical line to the file **/etc/passwd** on the NIS master server, and run **make**(1) in the directory **/var/yp** (see **ypmake**(8) for details) to propagate the change.

The new user is assigned a group and user ID number (GID and UID respectively). UIDs should be unique for each user and consistent across the NFS domain, since they control access to files. GIDs need not be unique. Typically, users working on similar projects will assigned to the same group. The system staff is group 10 for historical reasons, and the super-user is in this group.

An entry for a new user **francine** would look like this:

francine::235:20:& Featherstonehaugh:/usr/francine:/bin/csh

Fields in each password-file entry are delimited by colons, and have the following meanings:

- Login name (**francine**). The login name is limited to eight characters in length.
- Encrypted password or the string *##name* if encrypted passwords are stored in the password adjunct file. Typically, if passwords are to be stored in the main password file, this field is left empty, so no password is needed when the user first logs in. If security demands a password, it should be assigned by running **passwd**(1) immediately after exiting the editor. The number of significant characters in a password is eight. (See **passwd**(1).)
- User ID. The UID is a number which identifies that user uniquely in the system. Files owned by the user have this number stored in their data blocks, and commands such as **ls** (1V) (see **ls**(1V)), use it to look up the owner's login name. For this reason, you cannot randomly change this number. See **passwd**(5) for more information.
- Group ID. The GID number identifies the group to which the user belongs by default (although the user may belong to additional groups as well). All files that the user creates have this number stored in their data blocks, and commands such as **ls**(1V) (see **ls**(1V), use it to look up the group name. Group names and assignments are listed in the file **/etc/group** (which is described in **group**(5)) or in the NIS group map.
- This field is called the GCOS field (from earlier implementation of the operating system) and is traditionally used to hold the user's full name. Some installations have other information encoded in this field. From this information we can tell that Francine's real name is 'Francine Featherstonehaugh'. The & in the entry is shorthand for the user's login name.

- User's home directory. This is the directory in which that user is "positioned" when they log in.
- Initial shell which this user will see on login. If this field is empty, **sh**(1) is used as the initial shell.

An entry for a new user **francine** would look like this:

francine:::::lo:ad,+dw

Fields in each password adjunct file entry are delimited by colons, and have the following meanings:

- Login name (**francine**). This name must match the login name in the password file.
- Encrypted password. Typically, this field is left empty when adding the line using the editor. **passwd**(1) should be run immediately after exiting the editor.
- The next three fields are the minimum label, the maximum label, and the default label. These fields should be left empty, since they are reserved for future use.
- The next two fields are for the always-audit flags and the never-audit flags. Always-audit flags specify which events are guaranteed to be audited for that user. Never-audit flags specify which events are guaranteed not to be audited for that user. For a description of audit flags, see **audit_data**(5).

Making a Home Directory

As shown in the password file entry above, the name of Francine's home directory is to be **/usr/francine**. This directory must be created using **mkdir**(1), and Francine must be given ownership of it using **chown**(8), in order for her profile files to be read and executed, and to have control over access to it by other users:

```
example# mkdir /usr/francine
example# /usr/etc/chown francine /usr/francine
```

If running under NFS, the **mkdir**(1) and **chown**(8) commands must be performed on the NFS server.

Setting Up Skeletal Profile Files

New users often need assistance in setting up their profile files to initialize the terminal properly, configure their search path, and perform other desired functions at startup. Providing them with skeletal profile files saves time and interruptions for both the new user and the system administrator.

Such files as **.profile** (if they use **/usr/bin/sh** as the shell), or **.cshrc** and **.login** (if they use **/usr/bin/csh** as the shell), can include commands that are performed automatically at each login, or whenever a shell is invoked, such as **tset**(1). The ownership of these files must be changed to belong to the new user, either by running **su**(1V) before making copies, or by using **chown**(8).

FILES

/etc/passwd	password file
/etc/security/passwd.adjunct	
/etc/group	group file
/etc/yp/src/passwd	
~/.cshrc	
~/.login	
~/.profile	

SEE ALSO

csh(1), **ls**(1V), **make**(1), **mkdir**(1), **passwd**(1), **sh**(1), **su**(1V), **tset**(1), **audit**(2), **audit_control**(5), **audit_data**(5), **passwd.adjunct**(5), **group**(5), **passwd**(5), **passwd.adjunct**(5) **audit**(8), **auditd**(8), **chown**(8), **vipw**(8), **ypmake**(8)

System and Network Administration

NOTES

The Network Information Service (NIS) was formerly known as Sun Yellow Pages (YP). The functionality of the two remains the same; only the name has changed.

NAME

adv – advertise a directory for remote access with RFS

SYNOPSIS

adv
adv [**–r**] [**–d** *description*] *resource pathname* [*clients* ...]
adv –m *resource* **–d** *description* | [*clients*...]
adv –m *resource* [**–d** *description*] | *clients* ...

AVAILABILITY

This program is available with the *RFS* software installation option. Refer to *Installing SunOS 4.1* for information on how to install optional software.

DESCRIPTION

adv makes a resource from one system available for use on other systems. The machine that advertises the resource is called the server, while systems that mount and use the resource are **clients**. See **mount**(8). *resource* represents a directory, which could contain files, subdirectories, named pipes and devices.

Remote File Sharing (RFS) must be running before **adv** can be used to advertise or modify a resource entry.

When used with no options, **adv** displays all local resources that have been advertised; this includes the resource name, the pathname, the description, the read-write status, and the list of authorized clients. The resource field has a fixed length of 14 characters; all others are of variable length. Fields are separated by two SPACE characters and double quotes (") surround the description.

This command may be used without options by any user; otherwise it is restricted to the super-user.

There are three ways **adv** is used:

- To print a list of all locally-advertised resources, as shown by the first synopsis.
- To advertise the directory *pathname* under the name *resource* so it is available to RFS *clients*, as shown by the second synopsis.
- To modify *client* and *description* fields for currently advertised resources, as shown by the third and fourth synopses.

If any of the following are true, an error message will be sent to standard error.

- The network is not up and running.
- *pathname* is not a directory.
- *pathname* is not on a file system mounted locally.
- There is at least one entry in the *clients* field but none are syntactically valid.

OPTIONS

–r Restrict access to the resource to a read-only basis. The default is read-write access.

–d *description* Provide brief textual information about the advertised resource. *description* is a single argument surrounded by double quotes ("*argument*") and has a maximum length of 32 characters.

–m *resource* Modify information for a resource that has already been advertised. The resource is identified by a *resource* name. Only the *clients* and *description* fields can be modified. To change the *pathname*, *resource* name, or read/write permissions, you must unadvertise and re-advertise the resource.

resource This is the symbolic name used by the server and all authorized clients to identify the resource. It is limited to a maximum of 14 characters and must be different from every other resource name in the domain. All characters must be printable ASCII characters, but must not include '.' (periods), '/' (slashes), or white space.

pathname This is the local pathname of the advertised resource. It is limited to a maximum of 64 characters. This pathname cannot be the mount point of a remote resource and it can only be advertised under one resource name.

clients These are the names of all clients that are authorized to remotely mount the resource. The default is that all machines that can connect to the server are authorized to access the resource. Valid input is of the form *nodename*, *domain.nodename*, *domain.*, or an alias that represents a list of client names. A domain name must be followed by a '.' to distinguish it from a host name. The aliases are defined in **/etc/host.alias** and must conform to the alias capability in **mail**(1).

EXAMPLES

The following example displays the local resources that have been advertised:

```
example% adv
LOCAL_SUN3     /export/local/sun3 "" read-only unrestricted
LOCAL_SUN4     /export/local/sun4 "" read-only unrestricted
LOCAL_SHARE    /export/local/share "" read-only unrestricted
```

EXIT STATUS

If there is at least one syntactically valid entry in the *clients* field, a warning will be issued for each invalid entry and the command will return a successful exit status. A non-zero exit status will be returned if the command fails.

FILES

/etc/host.alias

SEE ALSO

mount(8), **rfstart**(8), **unadv**(8)

NAME

arp – address resolution display and control

SYNOPSIS

arp *hostname*

arp –a [*vmunix* [*kmem*]]

arp –d *hostname*

arp –s *hostname ether_address* [**temp**] [**pub**] [**trail**]

arp –f *filename*

DESCRIPTION

The **arp** program displays and modifies the Internet-to-Ethernet address translation tables used by the address resolution protocol (**arp**(4P)).

With no flags, the program displays the current ARP entry for *hostname*. The host may be specified by name or by number, using Internet dot notation.

OPTIONS

–a Display all of the current ARP entries by reading the table from the file *kmem* (default **/dev/kmem**) based on the kernel file *vmunix* (default **/vmunix**).

–d Delete an entry for the host called *hostname*. This option may only be used by the super-user.

–s Create an ARP entry for the host called *hostname* with the Ethernet address *ether_address*. The Ethernet address is given as six hex bytes separated by colons. The entry will be permanent unless the word **temp** is given in the command. If the word **pub** is given, the entry will be published, for instance, this system will respond to ARP requests for *hostname* even though the hostname is not its own. The word **trail** indicates that trailer encapsulations may be sent to this host.

–f Read the file named *filename* and set multiple entries in the ARP tables. Entries in the file should be of the form

hostname ether_address [**temp**] [**pub**] [**trail**]

with argument meanings as given above.

SEE ALSO

arp(4P), **ifconfig**(8C)

NAME

audit – audit trail maintenance

SYNOPSIS

audit [**–n** | **–s** | **–t**]
audit **–d** *username*
audit **–u** *username audit_event_state*

AVAILABILITY

This program is available with the *Security* software installation option. Refer to *Installing SunOS 4.1* for information on how to install optional software.

DESCRIPTION

The **audit** command is the general administrator's interface to kernel auditing. The process audit state for a user can be temporarily or permanently altered. The audit daemon may be notified to read the contents of the **audit_control** file and re-initialize the current audit directory to the first directory listed in the **audit_control** file, or to open a new audit file in the current audit directory specified in the **audit_control** file as last read by the audit daemon. Auditing may also be terminated/disabled.

OPTIONS

–n Signal audit daemon to close the current audit file and open a new audit file in the current audit directory.

–s Signal audit daemon to read audit control file. The audit daemon stores the information internally.

–t Signal audit daemon to disable auditing and die.

–d *username*
Change the process audit state of all processes owned by *username*. This new process audit state is constructed from the system and user audit values as specified in the **audit_control** and **passwd.adjunct** files respectively.

–u *username audit_event_state*
Set the process audit state from *audit_event_state* for all current processes owned by *username*. See **audit_control**(5) for the format of the system audit value. The process audit state is one argument. Enclose the audit event state in quotes, or do not use SPACE characters in the process audit state specification. A new login session reconstructs the process audit state from the audit flags in the **audit_control** and **passwd.adjunct** files.

SEE ALSO

audit(2), **setuseraudit**(2), **getauditflags**(3), **getfauditflags**(3), **audit_control**(5), **passwd.adjunct**(5)

NAME

auditd – audit daemon

SYNOPSIS

/usr/etc/auditd

DESCRIPTION

The audit daemon controls the generation and location of audit trail files. If the function **issecure**(3) returns false, the only action that **auditd** takes is to disable the auditing system; otherwise, auditing is set up and started. If auditing is desired, **auditd** reads the **audit_control**(5) file to get a list of directories into which audit files can be written and the percentage limit for how much space to reserve on each filesystem before changing to the next directory.

If **auditd** receives the signal **SIGUSR1**, the current audit file is closed and another is opened. If **SIGHUP** is received, the current audit trail is closed, the **audit_control** file reread, and a new trail is opened. If **SIGTERM** is received, the audit trail is closed and auditing is terminated. The program **audit**(8) sends these signals and is recommended for this purpose.

Each time the audit daemon opens a new audit trail file, it updates the file **audit_data**(5) to include the correct name.

Auditing Conditions

The audit daemon invokes the program **audit_warn**(8) under the following conditions with the indicated options:

audit_warn soft *pathname*

The file system upon which *pathname* resides has exceeded the minimum free space limit defined in **audit_control**(5). A new audit trail has been opened on another file system.

audit_warn allsoft

All available file systems have been filled beyond the minimum free space limit. A new audit trail has been opened anyway.

audit_warn hard *pathname*

The file system upon which *pathname* resides has filled or for some reason become unavailable. A new audit trail has been opened on another file system.

audit_warn allhard *count*

All available file systems have been filled or for some reason become unavailable. The audit daemon will repeat this call to **audit_warn** every twenty seconds until space becomes available. *count* is the number of times that **audit_warn** has been called since the problem arose.

audit_warn ebusy

There is already an audit daemon running.

audit_warn tmpfile

The file **/etc/security/audit/audit_tmp** exists, indicating a fatal error.

audit_warn nostart

The internal system audit condition is AUC_FCHDONE. Auditing cannot be started without rebooting the system.

audit_warn auditoff

The internal system audit condition has been changed to not be AUC_AUDITING by someone other than the audit daemon. This causes the audit daemon to exit.

audit_warn postsigterm

An error occurred during the orderly shutdown of the auditing system.

audit_warn getacdir

There is a problem getting the directory list from **/etc/security/audit/audit_control**.

The audit daemon will hang in a sleep loop until this file is fixed.

FILES

/etc/security/audit/audit_control
/etc/security/audit/audit_data

SEE ALSO

auditsvc(2), **audit_control**(5), **audit.log**(5), **audit**(8), **audit_warn**(8)

NAME

audit_warn – audit daemon warning script

SYNOPSIS

/usr/etc/audit_warn [*option* [*arguments*]]

DESCRIPTION

The **audit_warn** script processes warning or error messages from the audit daemon. When a problem is encountered, the audit daemon, **auditd**(8) calls **audit_warn** with the appropriate arguments. The *option* argument specifies the error type.

The system administrator can specify a list of mail recpients using the script's RECIPIENTS variable. The default recipient is root.

OPTIONS

soft *filename*
: indicates that the soft limit for *filename* has been exceeded. The default action for this option is to send mail to the system administrator.

allsoft indicates that the soft limit for all filesystems has been exceeded. The default action for this option is to send mail to the system administrator.

hard *filename*
: indicates that the hard limit for the file has been exceeded. The default action for this option is to send mail to the system administrator.

allhard *count*
: indicates that the hard limit for all filesystems has been exceeded *count* times. The default action for this option is to send mail to the system administrator only if the *count* is 1, and to send a message to console every time. It is recommended that mail *not* be send every time.

ebusy indicates that the audit daemon is already running. The default action for this option is to send mail to the system administrator.

tmpfile indicates that the temporary audit file already exists indicating a fatal error. The default action for this option is to send mail to the system administrator.

nostart indicates that auditing cannot be started because the system audit state is AUC_FCHDONE. The default action for this option is to send mail to the system administrator. Some system administrators may prefer to have the script reboot the system at this point.

auditoff
: indicates that someone other than the audit daemon changed the system audit state to something other than AUC_AUDITING. The audit daemon will have exited in this case. The default action for this option is to send mail to the system administrator.

postsigterm
: indicates that an error occurred during the orderly shutdown of the audit daemon. The default action for this option is to send mail to the system administrator.

getacdir
: indicates that there is a problem getting the directory list from: **/etc/security/audit/audit_control.**

The audit daemon will hang in a sleep loop until the file is fixed.

SEE ALSO

audit.log(5), **audit_control**(5), **audit**(8), **auditd**(8)

NAME

automount – automatically mount NFS file systems

SYNOPSIS

automount [**–mnTv**] [**–D** *name= value*] [**–f** *master-file*] [**–M** *mount-directory*] [**–tl** *duration*] [**–tm** *interval*] [**–tw** *interval*] [*directory map* [*–mount-options*]] ...

DESCRIPTION

automount is a daemon that automatically and transparently mounts an NFS file system as needed. It monitors attempts to access directories that are associated with an **automount** map, along with any directories or files that reside under them. When a file is to be accessed, the daemon mounts the appropriate NFS file system. You can assign a map to a directory using an entry in a direct **automount** map, or by specifying an indirect map on the command line.

The **automount** daemon appears to be an NFS server to the kernel. **automount** uses the map to locate an appropriate NFS file server, exported file system, and mount options. It then mounts the file system in a temporary location, and creates a symbolic link to the temporary location. If the file system is not accessed within an appropriate interval (five minutes by default), the daemon unmounts the file system and removes the symbolic link. If the indicated directory has not already been created, the daemon creates it, and then removes it upon exiting.

Since the name-to-location binding is dynamic, updates to an **automount** map are transparent to the user. This obviates the need to "pre-mount" shared file systems for applications that have "hard coded" references to files.

If the *directory* argument is a pathname, the *map* argument must be an *indirect* map. In an indirect map the key for each entry is a simple name that represents a symbolic link within *directory* to an NFS mount point.

If the *directory* argument is '/–', the map that follows must be a *direct* map. A direct map is not associated with a single directory. Instead, the key for each entry is a full pathname that will itself appear to be a symbolic link to an NFS mount point.

A map can be a file or a Network Interface Service (NIS) map; if a file, the *map* argument must be a full pathname.

The *–mount-options* argument, when supplied, is a comma-separated list of **mount**(8) options, preceded by a '–'. If these options are supplied, they become the default mount options for all entries in the map. Mount options provided within a map entry override these defaults.

OPTIONS

–m Suppress initialization of *directory-map* pairs listed in the **auto.master** NIS database.

–n Disable dynamic mounts. With this option, references through the **automount** daemon only succeed when the target filesystem has been previously mounted. This can be used to prevent NFS servers from cross-mounting each other.

–T Trace. Expand each NFS call and display it on the standard output.

–v Verbose. Log status and/or warning messages to the console.

–D *envar=value*
Assign *value* to the indicated **automount** (environment) variable.

–f *master-file*
Read a local file for initialization, ahead of the **auto.master** NIS map.

–M *mount-directory*
Mount temporary file systems in the named directory, instead of **/tmp_mnt**.

–tl *duration*
Specify a *duration*, in seconds, that a file system is to remain mounted when not in use. The default is 5 minutes.

-tm *interval*

Specify an *interval*, in seconds, between attempts to mount a filesystem. The default is 30 seconds.

-tw *interval*

Specify an *interval*, in seconds, between attempts to unmount filesystems that have exceeded their cached times. The default is 1 minute.

ENVIRONMENT

Environment variables can be used within an **automount** map. For instance, if **$HOME** appeared within a map, **automount** would expand it to its current value for the **HOME** variable. Environment variables are expanded only for the automounter's environment — not for the environment of a user using the automounter's services.

The special reference to **$ARCH** expands to the output of **arch (1)**. This can be useful in creating a map entry for mounting executables using a server's export pathname that varies according to the architecture of the client reading the map.

If a reference needs to be protected from affixed characters, you can surround the variable name with curly braces.

USAGE

Map Entry Format

A simple map entry (mapping) takes the form:

key [*-mount-options*] *location* ...

where *key* is the full pathname of the directory to mount when used in a direct map, or simple name in an indirect map. *mount-options* is a comma-separated list of **mount** options, and *location* specifies a remote filesystem from which the directory may be mounted. In the simple case, *location* takes the form:

hostname:*pathname*

Replicated Filesystems

Multiple *location* fields can be specified for replicated read-only filesystems, in which case **automount** sends multiple **mount** requests; **automount** mounts the file system from the first host that replies to the **mount** request. This request is first made to the local net or subnet. If there is no response, any connected server may respond. Since **automount** does not monitor the status of the server while the filesystem is mounted it will not use another location in the list if the currently mounted server crashes. This support for replicated filesystems is available only at mount time.

If each *location* in the list shares the same *pathname* then a single *location* may be used with a comma-separated list of hostnames.

hostname,*hostname* ... : *pathname*

Sharing Mounts

If *location* is specified in the form:

hostname:*pathname*:*subdir*

hostname is the name of the server from which to mount the file system, *pathname* is the pathname of the directory to mount, and *subdir*, when supplied, is the name of a subdirectory to which the symbolic link is made. This can be used to prevent duplicate mounts when multiple directories in the same remote file system may be accessed. With a map for **/home** such as:

able **homeboy:/home/homeboy:able**
baker **homeboy:/home/homeboy:baker**

and a user attempting to access a file in **/home/able**, **automount** mounts **homeboy:/home/homeboy**, but creates a symbolic link called **/home/able** to the **able** subdirectory in the temporarily-mounted filesystem. If a user immediately tries to access a file in **/home/baker**, **automount** needs only to create a symbolic link that points to the **baker** subdirectory; **/home/homeboy** is already mounted.

With the following map:

```
able    homeboy:/home/homeboy/able
baker   homeboy:/home/homeboy/baker
```

automount would have to mount the filesystem twice.

Comments and Quoting

A mapping can be continued across input lines by escaping the NEWLINE with a backslash. Comments begin with a # and end at the subsequent NEWLINE.

Characters that have special significance to the **automount** map parser may be protected either with double quotes (") or by escaping with a backslash (\). Pathnames with embedded whitespace, colons (:) or dollar ($) should be protected.

Directory Pattern Matching

The '&' character is expanded to the value of the *key* field for the entry in which it occurs. In this case:

```
able    homeboy:/home/homeboy:&
```

the & expands to **able**.

The '*' character, when supplied as the *key* field, is recognized as the catch-all entry. Such an entry will be used if any previous entry has not successfully matched the key being searched for. For instance, if the following entry appeared in the indirect map for **/home**:

```
*       &:/home/&
```

this would allow automatic mounts in **/home** of any remote file system whose location could be specified as:

hostname:/home/*hostname*

Multiple Mounts

A multiple mount entry takes the form:

key [/[*mountpoint* [*–mount-options*] *location* ...] ...

The initial / within the '/[*mountpoint*]' is required; the optional *mountpoint* is taken as a pathname relative to the destination of the symbolic link for *key*. If *mountpoint* is omitted in the first occurrence, a *mountpoint* of / is implied.

Given the direct map entry:

```
/arch/src  \
        /           –ro,intr    arch:/arch/src          alt:/arch/src  \
        /1.0        –ro,intr    alt:/arch/src/1.0       arch:/arch/src/1.0  \
        /1.0/man    –ro,intr    arch:/arch/src/1.0/man  alt:/arch/src/1.0/man
```

automount would automatically mount **/arch/src**, **/arch/src/1.0** and **/arch/src/1.0/man**, as needed, from either **arch** or **alt**, whichever host responded first. If the mounts are hierarchically related mounts closer to the root must appear before submounts. All the mounts of a multiple mount entry will occur together and will be unmounted together. This is important if the filesystems reference each other with relative symbolic links. Multiple mount entries can be used both in direct maps and in indirect maps.

Included Maps

The contents of another map can be included within a map with an entry of the form:

+mapname

mapname can either be a filename, or the name of an NIS map, or one of the special maps described below. If the key being searched for is not located in an included map, the search continues with the next entry.

Special Maps

There are two special maps currently available: **–hosts**, and **–null**. The **–hosts** map uses the NIS **hosts.byname** map to locate a remote host when the hostname is specified. This map specifies mounts of all exported file systems from any host. For instance, if the following **automount** command is already in effect:

automount /net –hosts

then a reference to **/net/hermes/usr** would initiate an automatic mount of all file systems from **hermes** that **automount** can mount; references to a directory under **/net/hermes** will refer to the corresponding directory relative to **hermes** root.

The **–null** map, when indicated on the command line, cancels any subsequent map for the directory indicated. It can be used to cancel a map given in **auto.master** or for a mount point specified as an entry in a direct map.

Configuration and the auto.master Map

automount normally consults the **auto.master** NIS configuration map for a list of initial **automount** maps, and sets up automatic mounts for them in addition to those given on the command line. If there are duplications, the command-line arguments take precedence over a local **–f** master map and they both take precedence over an NIS **auto.master** map. This configuration database contains arguments to the **automount** command, rather than mappings; unless **–f** is in effect, **automount** does *not* look for an **auto.master** file on the local host.

Maps given on the command line, or those given in a local **auto.master** file specified with **–f** override those in the NIS **auto.master** map. For instance, given the command:

automount –f /etc/auto.master /home –null /– /etc/auto.direct

and a file named **/etc/auto.master** that contains:

/home auto.home

automount would ignore /home entry in /etc/auto.master.

FILES

/tmp_mnt	directory under which filesystems are dynamically mounted

SEE ALSO

df(1V), **ls**(1V), **stat**(2V), **passwd**(5), **mount**(8)

System and Network Administration

NOTES

The **–hosts** map must mount all the exported filesystems from a server. If frequent access to just a single filesystem is required it is more efficient to access the filesystem with a map entry that is tailored to mount just the filesystem of interest.

When it receives signal number 1, **SIGHUP**, **automount** rereads the **/etc/mtab** file to update its internal record of currently-mounted file systems. If a file system mounted with **automount** is unmounted by a **umount** command, **automount** should be forced to reread the file.

An **ls**(1V) listing of the entries in the directory for an indirect map shows only the symbolic links for currently mounted filesystems. This restriction is intended to avoid unnecessary mounts as a side effect of programs that read the directory and **stat**(2V) each of the names.

Mount points for a single automounter must not be hierarchically related. **automount** will not allow an automount mount point to be created within an automounted filesystem.

automount must not be terminated with the **SIGKILL** signal (**kill –9**). Without an opportunity to unmount itself, the **automount** mount points will appear to the kernel to belong to a non-responding NFS server. The recommended way to terminate **automount** services is to send a **SIGTERM** (**kill –15**) signal to the daemon. This allows the automounter to catch the signal and unmount not only its daemon but also any mounts in **/tmp_mnt**. Mounts in **/tmp_mnt** that are busy will not be unmounted.

Since each direct map entry results in a separate mount for the mount daemon such maps should be kept short. Entries added to a direct map will have no effect until the automounter is restarted.

Entries in both direct and indirect maps can be modified at any time. The new information will be used when **automount** next uses the map entry to do a mount. **automount** does not cache map entries.

The Network Information Service (NIS) was formerly known as Sun Yellow Pages (YP). The functionality of the two remains the same; only the name has changed.

BUGS

The **bg** mount option is not recognized by the automounter.

Since **automount** is single-threaded, any request that is delayed by a slow or non-responding NFS server will delay all subsequent automatic mount requests until it completes.

Programs that read **/etc/mtab** and then touch files that reside under automatic mount points will introduce further entries to the file.

Automatically-mounted file systems are mounted with type **ignore**; they do not appear in the output of either **mount**(8), or **df**(1V).

NAME

boot – start the system kernel, or a standalone program

SYNOPSIS

>b [*device* [(*c*,*u*,*p*)]] [*filename*] [**–av**] *boot-flags*
>b?
>b!

DESCRIPTION

The boot program is started by the PROM monitor and loads the kernel, or another executable program, into memory.

The form **b?** displays all boot devices and their device arguments.

The form **b!** boots, but does not perform a RESET.

USAGE

Booting Standalone

When booting standalone, the boot program (**/boot**) is brought in by the PROM from the file system. This program contains drivers for all devices.

Booting a Sun-3 System Over the Network

When booting over the network, the Sun-3 system PROM obtains a version of the boot program from a server using the Trivial File Transfer Protocol (TFTP). The client broadcasts a RARP request containing its Ethernet address. A server responds with the client's Internet address. The client then sends a TFTP request for its boot program to that server (or if that fails, it broadcasts the request). The filename requested (unqualified — not a pathname) is the hexadecimal, uppercase representation of the client's Internet address, for example:

Using IP Address 192.9.1.17 = C0090111

When the Sun server receives the request, it looks in the directory **/tftpboot** for *filename*. That file is typically a symbolic link to the client's boot program, normally **boot.sun3** in the same directory. The server invokes the TFTP server, **tftpd**(8C), to transfer the file to the client.

When the file is successfully read in by the client, the boot program jumps to the load-point and loads **vmunix** (or a standalone program). In order to do this, the boot program makes a broadcast RARP request to find the client's IP address, and then makes a second broadcast request to a **bootparamd**(8) bootparams daemon, for information necessary to boot the client. The bootparams daemon obtains this information either from a local **/etc/bootparams** database file, or from a Network Interface Service (NIS) map. The boot program sends two requests to the bootparams daemon, the first, **whoami**, to obtain its hostname, and the second, **getfile**, to obtain the name of the client's server and the pathname of the client's root partition.

The boot program then performs a **mount**(8) operation to mount the client's root partition, after which it can read in and execute any program within that partition by pathname (including a symbolic link to another file within that same partition). Typically, it reads in the file **/vmunix**. If the program is not read in successfully, **boot** responds with a short diagnostic message.

Booting a Sun-2, Sun-4, or Sun386i System Over the Network

Sun-2, Sun-4 and Sun386i systems boot over the network in a similar fashion. However, the filename requested from a server must have a suffix that reflects the system architecture of the machine being booted. For these systems, the requested filename has the form:

ip-address.arch

where *ip-address* is the machine's Internet Protocol (IP) address in hex, and *arch* is a suffix representing its architecture. (Only Sun-3 systems may omit the *arch* suffix.) These filenames are restricted to 14 characters for compatibility with System V and other operating systems. Therefore, the architecture suffix is limited to 5 characters; it must be in upper case. At present, the following suffixes are recognized: SUN2 for Sun-2 system, SUN3 for Sun-3 system, SUN4 for Sun-4 system, S386 for Sun386i system, and PCNFS for PC-NFS. That file is typically a symbolic link to the client's boot program, normally **boot.sun2** in the same directory for a Sun-2 system, **boot.sun3** in the same directory for a Sun-3 system, or **boot.sun4** in the same directory for a Sun-4 system.

Note: a Sun-2 system boots from its server using one extra step. It broadcasts an ND request which is intercepted by the user-level **ndbootd** (8C) (see **ndbootd**(8C) server. This server sends back a standalone program that carries out the same TFTP request sequence as is done for all the other systems.

System Startup

Once the system is loaded and running, the kernel performs some internal housekeeping, configures its device drivers, and allocates its internal tables and buffers. The kernel then starts process number 1 to run **init**(8), which performs file system housekeeping, starts system daemons, initializes the system console, and begins multiuser operation. Some of these activities are omitted when **init** is invoked with certain *boot-flags*. These are typically entered as arguments to the boot command, and passed along by the kernel to **init**.

OPTIONS

device One of:

ie	Intel Ethernet
ec	3Com Ethernet
le	Lance Ethernet
sd	SCSI disk
st	SCSI 1/4" tape
mt	Tape Master 9-track 1/2" tape
xt	Xylogics 1/2" tape
xy	Xylogics 440/450/451 disk

c Controller number, **0** if there is only one controller for the indicated type of device.

u Unit number, **0** if only there is only one driver.

filename Name of a standalone program in the selected partition, such as **stand/diag** or **vmunix**. Note: *filename* is relative to the root of the selected device and partition. It never begins with '/' (slash). If *filename* is not given, the boot program uses a default value (normally **vmunix**). This is stored in the **vmunix** variable in the **boot** executable file supplied by Sun, but can be patched to indicate another standalone program loaded using **adb**(1).

–a Prompt interactively for the device and name of the file to boot. For more information on how to boot from a specific device, refer to *Installing SunOS 4.1*.

–v Verbose. Print more detailed information to assist in diagnosing diskless booting problems.

boot-flags The boot program passes all *boot-flags* to the kernel or standalone program. They are typically arguments to that program or, as with those listed below, arguments to programs that it invokes.

–b Pass the **–b** flag through the kernel to **init**(8) so as to skip execution of the **/etc/rc.boot** script.

–h Halt after loading the system.

–s Pass the **–s** flag through the kernel to **init**(8) for single-user operation.

–i *initname*
Pass the **–i** *initname* to the kernel to tell it to run *initname* as the first program rather than the default **/sbin/init**.

FILES

/boot	standalone boot program
/tftpboot/*address*	symbolic link to the boot program for the client whose Internet address, in uppercase hexadecimal, is *address*
/tftpboot/boot.sun3	Sun-3 first stage boot program
/tftpboot/boot.sun4	Sun-4 first stage boot program
/usr/etc/in.tftpd	TFTP server
/usr/mdec/installboot	program to install boot blocks from a remote host
/vmunix	kernel file that is booted by default
/etc/bootparams	file defining root and swap paths for clients

SEE ALSO

adb(1), **tftp**(1C) **bootparamd**(8), **init**(8), **kadb**(8S), **monitor**(8S), **mount**(8), **ndbootd**(8C), **rc**(8), **reboot**(8), **tftpd**(8C)

Installing SunOS 4.1
System and Network Administration

BUGS

On Sun-2 systems, the PROM passes in the default name **vmunix**, overriding the the boot program's patchable default.

NOTES

The Network Information Service (NIS) was formerly known as Sun Yellow Pages (YP). The functionality of the two remains the same; only the name has changed.

NAME

bootparamd – boot parameter server

SYNOPSIS

/usr/etc/rpc.bootparamd [–d]

DESCRIPTION

bootparamd is a server process that provides information to diskless clients necessary for booting. It first consults the local **/etc/bootparams** file for a client entry. If the local bootparams file does not exist, **bootparamd** consults the corresponding Network Interface Service (NIS) map.

bootparamd can be invoked either by **inetd**(8C) or by the user.

OPTIONS

–d Display the debugging information.

FILES

/etc/bootparams

SEE ALSO

inetd(8C)

NOTES

The Network Information Service (NIS) was formerly known as Sun Yellow Pages (YP). The functionality of the two remains the same; only the name has changed.

NAME

C2conv, C2unconv – convert system to or from C2 security

SYNOPSIS

C2conv

C2unconv

AVAILABILITY

This program is available with the *Security* software installation option. Refer to *Installing SunOS 4.1* for information on how to install optional software.

DESCRIPTION

C2conv converts a standard SunOS system to operate with C2-level security.

The program prompts for information regarding the Secure NFS option, client systems (if the system is an NFS server for diskless clients), audit devices (if it is an audit file server), and names of file systems (if there is a remote audit server). The program also requests certain information for the **audit_control**(5) file; default values may be used for audit flags and for the "minfree" value. Finally, it requests the mail address to be used (by **mail**(1)) when C2 administrative tasks are required. The default address is **root** for the host being converted.

Once it has this information, **C2conv** uses it to set up the necessary files for a C2 secure system, reporting on its progress as it proceeds.

C2unconv backs out the changes made to **/etc/passwd** and **/etc/group**. It does not back out changes to other files.

FILES

/etc/passwd
/etc/group
/etc/fstab

SEE ALSO

audit_control(5)

NAME

captoinfo – convert a termcap description into a terminfo description

SYNOPSIS

captoinfo [–v . . .] [–V] [–1] [–w *width*] *filename*. . .

SYNOPSIS

/usr/5bin/captoinfo [–v . . .] [–V] [–1] [–w *width*] *filename*. . .

AVAILABILITY

The System V version of this command is available with the *System V* software installation option. Refer to *Installing SunOS 4.1* for information on how to install optional software.

DESCRIPTION

captoinfo converts the **termcap**(5) terminal description entries given in *filename* into **terminfo**(5V) source entries, and writes them to the standard output along with any comments found in that file. A description that is expressed as relative to another description (as specified in the **termcap** *tc*= capability) is reduced to the minimum superset before being written.

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

OPTIONS

–v Verbose. Print tracing information on the standard error as the program runs. Additional –v options increase the level of detail.

–V Version. Display the version of the program on the standard error and exit.

–1 Print fields one-per-line. Otherwise, fields are printed several to a line, to a maximum width of 60 characters.

–w *width*
Change the output to *width* characters.

FILES

/usr/share/lib/terminfo/?/* compiled terminal description database
/etc/termcap

SEE ALSO

curses(3V), **termcap**(5), **terminfo**(5V), **infocmp**(8V), **tic**(8V)

DIAGNOSTICS

tgetent failed with return code n
The termcap entry is not valid. In particular, check for an invalid '**tc=**' entry.

unknown type given for the termcap code *cc*.
The termcap description had an entry for *cc* whose type was not boolean, numeric or string.

wrong type given for the boolean (numeric, string) termcap code *cc*.
The boolean **termcap** entry *cc* was entered as a numeric or string capability.

the boolean (numeric, string) termcap code *cc* **is not a valid name.**
An unknown **termcap** code was specified.

tgetent failed on TERM=term.
The terminal type specified could not be found in the **termcap** file.

TERM=term: cap *cc* **(info** *ii***) is**
The **termcap** code was specified as a null string. The correct way to cancel an entry is with an '**@**', as in '**:bs@:**'. Giving a null string could cause incorrect assumptions to be made by the software which uses **termcap** or **terminfo**.

a function key for *cc* was specified, but it already has the value *vv*. When parsing the **ko** capability, the key *cc* was specified as having the same value as the capability *cc*, but the key *cc* already had a value assigned to it.

the unknown termcap name *cc* was specified in the ko termcap capability. A key was specified in the **ko** capability which could not be handled.

the *vi* character v (info *ii*) has the value *xx*, but ma gives n. The **ma** capability specified a function key with a value different from that specified in another setting of the same key.

the unknown *vi* key v was specified in the ma termcap capability. A vi(1) key unknown to **captoinfo** was specified in the **ma** capability.

Warning: termcap sg (*nn*) and termcap ug (*nn*) had different values. **terminfo** assumes that the **sg** (now **xmc**) and **ug** values were the same.

Warning: the string produced for *ii* may be inefficient. The parameterized string being created should be rewritten by hand.

Null termname given. The terminal type was null. This is given if the environment variable TERM is not set or is null.

cannot open *filename* for reading. The specified file could not be opened.

WARNINGS

Certain **termcap** defaults are assumed to be true. The bell character (**terminfo bel**) is assumed to be ^G. The linefeed capability (**termcap nl**) is assumed to be the same for both **cursor_down** and **scroll_forward** (**terminfo cud1** and **ind**, respectively.) Padding information is assumed to belong at the end of the string.

The algorithm used to expand parameterized information for **termcap** fields such as **cursor_position** (**termcap cm**, **terminfo cup**) can sometimes produce a string that may not be optimal. In particular, the rarely used **termcap** operation *%n* produces strings that are especially long. Most occurrences of these non-optimal strings will be flagged with a warning message and may need to be recoded by hand.

The short two-letter name at the beginning of the list of names in a **termcap** entry, a hold-over from an earlier version of the system, has been removed.

NAME

catman – create the cat files for the manual

SYNOPSIS

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

DESCRIPTION

catman creates the preformatted versions of the on-line manual from the **nroff**(1) input files. Each manual 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** database.

If there is one parameter not starting 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**.

If an unformatted source file contains only a line of the form '**.so manx/yyy.x**', a symbolic link is made in the catx or fmtx directory to the appropriate preformatted manual page. This feature allows easy distribution of the preformatted manual pages among a group of associated machines with **rdist**(1), since it makes the directories of preformatted manual pages self-contained and independent of the unformatted entries.

OPTIONS

–n Do not (re)create the **whatis** database.

–p Print what would be done instead of doing it.

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

–w Only create the **whatis** database that is used by **whatis**(1) and the **man**(1) **–f** and **–k** options. No manual reformatting is done.

–M Update manual pages located in the specified **directory** (**/usr/man** by default).

–T Use **tmac.an** in place of the standard manual page macros.

ENVIRONMENT

TROFF The name of the formatter to use when the **–t** flag is given. If not set, '**troff**' is used.

FILES

/usr/[share]/man	default manual directory location
/usr/[share]/man/man?/*.*	raw (nroff input) manual sections
/usr/[share]/man/cat?/*.*	preformatted **nroff**ed manual pages
/usr/[share]/man/fmt?/*.*	preformatted **troff**ed manual pages
/usr/[share]/man/whatis	**whatis** database location
/usr/lib/makewhatis	command script to make whatis database

SEE ALSO

apropos(1), **man**(1), **nroff**(1), **rdist**(1), **troff**(1), **whatis**(1)

NOTES

If the **–n** option is specified, the **/usr/man/whatis** database is not created and the **apropos**, **whatis**, '**man –f**', and '**man –k**' commands will fail.

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 directory **/usr/man**.

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.

NAME

change_login – control screen blanking and choice of login utility

SYNOPSIS

change_login

AVAILABILITY

Available only on Sun 386i systems running a SunOS 4.0.*x* release or earlier. Not a SunOS 4.1 release feature.

DESCRIPTION

To prolong the life of your monitor, your Sun386i system turns off the screen display if you have not used the keyboard or mouse for 30 minutes or more. To see the screen again, simply move the mouse on the pad or press any key. This feature is normally enabled automatically when you log in, but you can control it using the **change_login** command as explained below.

This command also determines whether you log into your workstation using the Sun386i login screen, **logintool**(8) or through a traditional **login:** prompt.

The screen blanking choices available with **change_login** are:

1. Logintool and Sun Logo screenblank
Enables screen blanking. When blank, the system displays the Sun logo moving randomly around an otherwise dark screen.

2. Logintool and video-off screenblank
Shuts off the video output to your monitor when the screen goes blank. This is the most efficient type of screen blanking. The Desktop is almost instantly redisplayed when you move the mouse or begin typing.

3. Logintool and no screenblank
Retains the login screen, but disables screen blanking.

4. No Logintool and no screenblank
Disables both the login screen and screen blanking.

EXAMPLE

The following is an example of **change_login**. Notice you must be super-user to use this command.

```
example# change_login

This program will check what login and screenblank
options are set on this workstation, and allow you
to choose other options, if you are logged in as superuser.

Do you want to do this? [y or n]: y

This workstation is set up to use logintool and
a screenblank program that displays a Sun logo graphic.

These are the available options:

+ 1. Logintool and Sun Logo screenblank
  2. Logintool and video-off screenblank
  3. Logintool and no screenblank
  4. No Logintool and no screenblank

+ indicates the current configuration

You must be logged in as superuser to change the current setting.
```

Follow the instructions in *Sun386i System Setup and Maintenance* or *Sun386i Advanced Administration* to shut down and then restart your system. The setting chosen in the above example will not be enabled until you have restarted your system.

SEE ALSO

login(1), **screenblank**(1), **su**(1V), **logintool**(8)

Sun386i Advanced Skills
Sun386i System Setup and Maintenance
Sun386i Advanced Administration

NAME

chown – change owner

SYNOPSIS

/usr/etc/chown [**–fHR**] *owner*[*.group*] *filename* ...

DESCRIPTION

chown changes the owner of the *filename*s to *owner*. The owner may be either a decimal user ID (UID) or a login name found in the password file. An optional *group* may also be specified. The group may be either a decimal group ID (GID) or a group name found in the GID file.

Only the super-user can change owner, in order to simplify accounting procedures.

OPTIONS

–f Do not report errors.

–R Recursively descend into directories setting the ownership of all files in each directory encountered. When symbolic links are encountered, their ownership is changed, but they are not traversed.

FILES

/etc/passwd password file

SEE ALSO

chgrp(1), **chown**(2V), **group**(5), **passwd**(5)

NAME

chroot – change root directory for a command

SYNOPSIS

/usr/etc/chroot *newroot command*

DESCRIPTION

The given command is executed *relative* to the new root. The meaning of any initial '/' (slashes) in path names is changed for a command and any of its children to *newroot*. Furthermore, the initial working directory is *newroot*.

Input and output redirections on the command line are made with respect to the *original* root:

chroot newroot command >x

creates the file *x* relative to the original root, not the new one.

This command is restricted to the super-user.

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

SEE ALSO

chdir(2V)

BUGS

One should exercise extreme caution when referring to special files in the new root file system.

NAME

chrtbl – generate character classification table

SYNOPSIS

/usr/etc/chrtbl [*filename*]

DESCRIPTION

chrtbl converts a source description of a character classification table into a form that can be used by the character classification functions and multibyte functions (see **ctype**(3V) and **mblen**(3)). The source description is found in *filename*. If *filename* is not given, or just given as '–', **chrtbl** reads its source description from the standard input.

chrtbl creates one or two output files, the second file is only created if the **model** token is specified. By default, these files are created in the current working directory. The first file, named by the **chrclass** token, is always produced and contains the character classification information for all single-byte (7-bit and 8-bit) character code-sets described by one setting of the LC_CTYPE category of locale. The second file, created if the **model** token is specified, contains information relating to details of width and structure of the coded character set currently under definition. The second file is named by appending '**.ci**'. to the value specified by the **chrclass** token.

The first output file contains a binary form of the character classification information described in *filename*. It is structured in such a way that it can be used at run-time to replace the active version of the **ctype**[ ] array in the C-library, For it to be understood at run-time, the output file must be moved to the **/usr/share/lib/locale/LC_TYPE** or **/etc/locale** directory (see FILES below) by the super-user or a member of group **bin**. This file must be readable by user, group, and other; no other permission should be set.

filename contains a sequence of tokens in any order after the **chrclass** token, each separated by one or more NEWLINE characters or comment lines. The tokens recognized by **chrtbl** are as follows:

chrclass *name*

name is the filename or pathname of the character classification file. This is a mandatory token. It must be the first token to be defined, and is usually given the name that relates to a valid setting of the LC_CTYPE category of locale.

model *name,args*

This optional token chooses the type of character code-set announcement mechanism associated with the character classification table generated by **chrtbl**. The name of the file created by this token is the name specified by the **chrclass** token, concatenated with a '**.ci**'. The arguments to **model** must be one of the following:

euc x,y,z

The model file contains information describing the required setting for the Extended Unix code-set announcement mechanism. *x,y,z* relate to the storage widths (in bytes) of EUC code-sets 1, 2 and 3 respectively.

xccs The model file contains information describing the Xerox Character Code Standard (XC1-3-3-0) announcement mechanism. There are no additional arguments required.

iso2022 g0,g1,g2,g3 x

The model file contains information describing a generative version of the ISO-2022 code set announcement mechanism. The multibyte functions driven by this model are capable of handling the standard one or more byte escape sequences as well as all of the standard shift functions. The four arguments *g0,g1,g2,g3* define the default width (in bytes) of the four designations (respectively) available under ISO-2022, Maximum integer value of any of these arguments is 2. The fianl argument *x* is mandatory and must be set to either 7 or 8. It selects the default bit-width of each byte on input and output to/from the multibyte functions.

If the **model** token is declared without arguments, then it is assumed that there is a set of user-defined rules for character code-set announcement. This is noted in the output file and will be later used to fold in user-defined code into the multibyte functions in the C-library (see **mblen**(3)).

isupper Character codes to be classified as upper-case letters.

islower Character codes to be classified as lower-case letters.

isdigit Character codes to be classified as numeric.

isspace Character codes to be classified as a spacing (delimiter) character.

ispunct Character codes to be classified as a punctuation character.

iscntrl Character codes to be classified as a control character.

isblank Character code for the space character.

isxdigit Character codes to be classified as hexadecimal digits.

ul Relationship between upper- and lower-case characters.

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

A character 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 tab characters.

The dash (–) 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 RETURN is permitted after the backslash character.

The relationship between upper- and lower-case letters (**ul**) is expressed as ordered pairs of octal and hexadecimal constants:

<upper-case_character lower-case_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.

EXAMPLES

The following is an example of an input file used to create the ASCII code set definition table on a file named **ascii**.

```
chrclass    ascii
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>
```

FILES

/usr/share/lib/locale/LC_CTYPE/*	run-time location of the character classification tables generated by **chrtbl**
/etc/locale/LC_CTYPE/*	location for private versions of the classification tables generated by **chrtbl**

SEE ALSO

ctype(3V), **environ**(5V)

DIAGNOSTICS

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

NAME

client – add or remove diskless Sun386i systems

SYNOPSIS

client [**–a** *arch*] [**–h** *hostid*] [**–o** *os*] [**–q**] [**–t** *minutes*] **add** *bootserver client etheraddress ipaddress*

client remove *client*

client modify *client* [**diskful** | **diskless** | **slave**]

AVAILABILITY

Available only on Sun 386i systems running a SunOS 4.0.*x* release or earlier. Not a SunOS 4.1 release feature.

DESCRIPTION

client can be used to manually add and remove diskless clients of a PNP boot server. After successful completion of the command, the diskless client can boot. Only users in the *networks* group (group 12) on the boot server are allowed to change configurations using this utility. **client** can be invoked from any system on the network.

The boot server of a system is the only machine truly required for that system to boot to the point of allowing user logins; it must accordingly provide name, booting, and time services. Diskless clients can provide none of these services themselves. Diskful clients, however can provide most of their own boot services. Network clients only need name and time services from the network, and can use any boot server.

To add a diskless client, use the **add** operation. To remove a diskless, diskful, or network client, use the **remove** operation. To change a system's network role, use the **modify** operation.

A server can reject a configuration request if it is disallowed by the contents of the **bootservers** map (e.g., too many clients would be configured, or too little free space would be left on the server), or if no system software for the client is available.

OPTIONS

–a *arch* Specifies the architecture code of the client; it defaults to **s386**. (Note: architecture codes are different from architecture names. Architecture codes are used in diskless booting, and are at most five characters in length, while architecture names can be longer.)

–h *hostid* Specifies the host ID of the client; if supplied, it is used as the root password for the system. It defaults to the null string.

–o *os* Specifies the operating system; defaults to 'unix'. This is currently used only to construct the system's *publickey* data, where applicable; this is never done if the system has no *hostid* specified.

–q Quiet. Displays only error messages.

–t *minutes* Sets the RPC timeout to the number of minutes indicated; this defaults to 15 minutes. If the bootserver takes more time than this to complete, **client** will exit. Unless the server has already completed setup, but not yet sent status to **client**, this will cause the bootserver to back out of the setup, deallocating all assigned resources.

SEE ALSO

publickey(5) **netconfig**(8C), **pnpd**(8C)

BUGS

Unless the *hostid* is assigned, the root filesystem for the diskless client is not set up beyond copying the **proto** and **boot** files into it. This means that **netconfig** will often handle other parts of the setup.

NAME

clri – clear inode

SYNOPSIS

/usr/etc/clri *filesystem i-number...*

DESCRIPTION

Note: **clri** has been superseded for normal file system repair work by **fsck**(8).

clri writes zeros on the inodes with the decimal *i-numbers* on the *filesystem*. After **clri,** any blocks in the affected file will show up as "missing" in an **icheck**(8) of the *filesystem.*

Read and write permission is required on the specified file system device. The inode becomes allocatable.

The primary purpose of this routine is to remove a file which for some reason appears in no directory. If it is used to zap an inode which does appear in a directory, care should be taken to track down the entry and remove it. Otherwise, when the inode is reallocated to some new file, the old entry will still point to that file. At that point removing the old entry will destroy the new file. The new entry will again point to an unallocated inode, so the whole cycle is likely to be repeated again and again.

SEE ALSO

icheck(8) **fsck**(8)

BUGS

If the file is open, **clri** is likely to be ineffective.

NAME

colldef – convert collation sequence source definition

SYNOPSIS

/usr/etc/colldef *filename*

DESCRIPTION

colldef converts a collation sequence source definition into a format usable by the **strxfrm( )** and **strcoll**(3) functions. It is used to define the many ways in which strings can be ordered and collated.

colldef reads the collation sequence source definition from the standard input and stores the converted definition in *filename*.

The collation sequence definition specifies a set of collating elements and the rules defining how strings containing these should be ordered. This is most useful for different language definitions. The rules provide the following capabilities:

1-to-Many mapping
: A single character is mapped into a string of collating elements.

Many-to-1 mapping
: A string of two or more characters is mapped as a single collating element.

Null string mapping
: A character, or string of characters, is mapped to a null collating element (that is, will be ignored).

Equivalence class definition.
: A collection of characters that have the same value.

Secondary ordering within equivalence class.

USAGE

The following keywords may be used in the input file *filename*.

charmap
: Optional keyword. Defines where a mapping of the character and collating element symbols to the actual character encoding can be found.

substitute
: Optional keyword. Defines a one-to-many mapping between a single byte and a character string.

order Mandatory keyword. Defines the primary and secondary ordering of collating elements within this collation table.

EXIT STATUS

colldef exits with the following values:

0 No errors were found and the output was successfully created.

>0 Errors were found.

FILES

/etc/locale/LC_COLLATE/*locale/domain*
: standard private location for collation orders under the *locale* locale

/usr/share/lib/locale/LC_COLLATE
: standard shared location for collation orders under the *locale* locale

SEE ALSO

strcoll(3)
System Services Overview

NAME

comsat, in.comsat – biff server

SYNOPSIS

/usr/etc/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**(8C), and times out if inactive for a few minutes.

comsat listens on a datagram port associated with the **biff**(1) service specification (see **services**(5)) 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.

FILES

/etc/utmp to find out who's logged on and on what terminals

SEE ALSO

biff(1), **services**(5), **inetd**(8C)

BUGS

The message header filtering is prone to error.

The notification should appear in a separate window so it does not mess up the screen.

NAME

config – build system configuration files

SYNOPSIS

/usr/etc/config [**–fgnp**] [**–o** *obj_dir*] *config_file*

DESCRIPTION

config does the preparation necessary for building a new system kernel with **make**(1). The *config_file* named on the command line describes the kernel to be made in terms of options you want in your system, size of tables, and device drivers to be included. When you run **config**, it uses several input files located in the current directory (typically the **conf** subdirectory of the system source including your *config_file*). The format of this file is described below.

If the directory named *../config_file* does not exist, **config** will create one. One of **config**'s output files is a makefile which you use with **make**(1) to build your system.

You use **config** as follows. Run **config** from the **conf** subdirectory of the system source (in a typical Sun environment, from **/usr/share/sys/sun[2 3 4]/conf**):

```
example# /usr/etc/config config_file
Doing a "make depend"
example# cd ../config_file
example# make
...lots of output...
```

While **config** is running watch for any errors. Never use a kernel which **config** has complained about; the results are unpredictable. If **config** completes successfully, you can change directory to the *../config_file* directory, where it has placed the new makefile, and use **make** to build a kernel. The output files placed in this directory include **ioconf.c**, which contains a description of I/O devices attached to the system; **mbglue.s**, which contains short assembly language routines used for vectored interrupts, a makefile, which is used by **make** to build the system; a set of header files (*device_name*.**h**) which contain the number of various devices that may be compiled into the system; and a set of swap configuration files which contain definitions for the disk areas to be used for the root file system, swapping, and system dumps.

Now you can install your new kernel and try it out.

OPTIONS

–f Set up the makefile for fast builds. This is done by building a **vmunix.o** file which includes all the **.o** files which have no source. This reduces the number of files which have to be **stat**ed during a system build. This is done by prelinking all the files for which no source exists into another file which is then linked in place of all these files when the kernel is made. This makefile is faster because it does not **stat** the object files during the build.

–g Get the current version of a missing source file from its SCCS history, if possible.

–n Do not do the '**make depend**'. Normally **config** will do the '**make depend**' automatically. If this option is used **config** will print '**Don't forget to do a "make depend"**' before completing as a reminder.

–p Configure the system for profiling (see **kgmon**(8) and **gprof**(1)).

–o *obj_dir*
Use *../obj_dir* instead of **../OBJ** as the directory to find the object files when the corresponding source file is not present in order to generate the files necessary to compile and link your kernel.

USAGE

Input Grammar

In the following descriptions, a number can be a decimal integer, a whole octal number or a whole hexadecimal number. Hex and octal numbers are specified to **config** in the same way they are specified to the C compiler, a number starting with **0x** is a hex number and a number starting with just a **0** is an octal number.

Comments are begin with a # character, and end at the next NEWLINE. Lines beginning with TAB characters are considered continuations of the previous line. Lines of the configuration file can be one of two basic types. First, there are lines which describe general things about your system:

machine "*type***"**

This is system is to run on the machine type specified. Only one machine type can appear in the config file. The legal *type*s for a Sun system are **sun2**, **sun3**, **sun4**, and **sun386**. Note: the double quotes around *type* are part of the syntax, and must be included.

cpu "*type***"**

This system is to run on the cpu type specified. More than one cpu type can appear in the config file. Legal *type*s for a **sun2** machine are noted in the annotated config file in *Installing SunOS 4.1*.

ident *name*

Give the system identifier — a name for the machine or machines that run this kernel. Note that *name* must be enclosed in double quotes if it contains both letters and digits. Also, note that if *name* is GENERIC, you need not include the '**options GENERIC**' clause in order to specify '**swap generic**'.

maxusers *number*

The maximum expected number of simultaneously active user on this system is *number*. This number is used to size several system data structures.

options *optlist*

Compile the listed options into the system. Options in this list are separated by commas. A line of the form:

options FUNNY,HAHA

yields

–DFUNNY –DHAHA

to the C compiler. An option may be given a value, by following its name with = (equal sign) then the value enclosed in (double) quotes. None of the standard options use such a value.

In addition, options can be used to bring in additional files if the option is listed in the **files** files. All options should be listed in upper case. In this case, no corresponding *option*.**h** will be created as it would be using the corresponding *pseudo-device* method.

config *sysname config_clauses* . . .

Generate a system with name *sysname* and configuration as specified in *config-clauses*. The *sysname* is used to name the resultant binary image and per-system swap configuration files. The *config_clauses* indicate the location for the root file system, one or more disk partitions for swapping and paging, and a disk partition to which system dumps should be made. All but the root device specification may be omitted; **config** will assign default values as described below.

root A root device specification is of the form '**root on** *xy0d*'. If a specific partition is omitted — for example, if only **root on xy0** is specified — the '**a**' partition is assumed. When a generic system is being built, no root specification should be given; the root device will be defined at boot time by prompting the console.

swap To specify a swap partition, use a clause of the form: '**swap on** *partition*'. Swapping areas may be almost any size. Partitions used for swapping are sized at boot time by the system; to override dynamic sizing of a swap area the number of sectors in the swap area can be specified in the config file. For example, '**swap on** *xy0b* **size 99999**' would configure a swap partition with 99999 sectors. If **swap generic** or no *partition* is specified with **on**, partition b on the root device is used. For dataless clients, use '**swap on type nfs**'.

To configure multiple swap partitions, specify multiple '**swap on**' clauses. For example:

config vmunix swap on xy0 swap on xy1

dumps The location to which system dumps are sent may be specified with a clause of the form '**dumps on** *xy1*'. If no dump device is specified, the first swap partition specified is used. If a device is specified without a particular partition, the '**b**' partition is assumed. If a generic configuration is to be built, no dump device should be specified; the dump device will be assigned to the swap device dynamically configured at boot time. Dumps are placed at the end of the partition specified. Their size and location is recorded in global kernel variables *dumpsize* and *dumplo*, respectively, for use by **savecore**(8).

Device names specified in configuration clauses are mapped to block device major numbers with the file **devices.***machine*, where *machine* is the machine type previously specified in the configuration file. If a device name to block device major number mapping must be overridden, a device specification may be given in the form '**major** *x* **minor** *y*'.

The second group of lines in the configuration file describe which devices your system has and what they are connected to (for example, a Xylogics 450 Disk Controller at address 0xee40 in the Multibus I/O space). These lines have the following format:

dev_type dev_name **at** *con_dev more_info*

dev_type is either **controller, disk, tape, device,** or **pseudo-device.** These types have the following meanings:

controller A disk or tape controller.

disk or **tape** Devices connected to a controller.

device Something "attached" to the main system bus, like a cartridge tape interface.

pseudo-device A software subsystem or driver treated like a device driver, but without any associated hardware. Current examples are the pseudo-tty driver and various network subsystems. For pseudo-devices, **more_info** may be specified as an integer, that gives the value of the symbol defined in the header file created for that device, and is generally used to indicate the number of instances of the pseudo-device to create.

dev_name is the standard device name and unit number (if the device is not a **pseudo-device**) of the device you are specifying. For example, **xyc0** is the *dev_name* for the first Xylogics controller in a system; **ar0** names the first quarter-inch tape controller.

con_dev is what the device you are specifying is connected to. It is either nexus?, a bus type, or a controller. There are several bus types which are used by **config** and the kernel.

The different possible bus types are:

obmem	On board memory
obio	On board io
mbmem	Multibus memory (**sun2** system only)
mbio	Multibus io (**sun2** system only)
vme16d16 (vme16)	16 bit VMEbus/ 16 bit data
vme24d16 (vme24)	24 bit VMEbus/ 16 bit data
vme32d16	32 bit VMEbus/ 16 bit data (**sun3** system only)
vme16d32	16 bit VMEbus/ 32 bit data (**sun3** system only)
vme24d32	24 bit VMEbus/ 32 bit data (**sun3** system only)
vme32d32 (vme32)	32 bit VMEbus/ 32 bit data (**sun3** system only)

All of these bus types are declared to be connected to nexus. The devices are hung off these buses. If the bus is wildcarded, then the autoconfiguration code will determine if it is appropriate to probe for the device on the machine that it is running on. If the bus is numbered, then the autoconfiguration code will only look for that device on machine type N. In general, the Multibus and VMEbus bus types are always wildcarded.

more_info is a sequence of the following:

csr *address* — Specify the address of the csr (command and status registers) for a device. The csr addresses specified for the device are the addresses within the bus type specified.

The csr address must be specified for all controllers, and for all devices connected to a main system bus.

drive *number* — For a disk or tape, specify which drive this is.

flags *number* — These flags are made available to the device driver, and are usually read at system initialization time.

priority *level* — For devices which interrupt, specify the interrupt level at which the device operates.

vector *intr number* [*intr number* . . .]
For devices which use vectored interrupts on VMEbus systems, *intr* specify the vectored interrupt routine and *number* the corresponding vector to be used (0x40-0xFF).

A **?** may be substituted for a number in two places and the system will figure out what to fill in for the **?** when it boots. You can put question marks on a *con_dev* (for example, at virtual '?'), or on a drive number (for example, drive '?'). This allows redundancy, as a single system can be built which will boot on different hardware configurations.

The easiest way to understand **config** files it to look at a working one and modify it to suit your system. Good examples are provided in *Installing SunOS 4.1*.

FILES

Files in **/usr/share/sys/sun[2 3 4]/conf** which may be useful for developing the *config_file* used by **config** are:

GENERIC — These are generic configuration files for either a Sun-2 or Sun-3 system. They contain all possible device descriptions lines for the particular architecture.

README — File describing how to make a new kernel.

As shipped from Sun, the files used by **/usr/etc/config** as input are in the **/usr/include/sys/conf** directory:

config_file	System-specific configuration file
Makefile.src	Generic prototype makefile for Sun–[23] systems
files	List of common files required to build a basic kernel
devices	Name to major device mapping file for Sun–[23] systems

/usr/etc/config places its output files in the *../config_file* directory:

mbglue.s	Short assembly language routines used for vectored interrupts
ioconf.c	Describes I/O devices attached to the system
makefile	Used with **make**(1) to build the system
device_name.**h**	a set of header files (various *device_name*'s) containing devices which can be compiled into the system

SEE ALSO

gprof(1), **make**(1), **kgmon**(8), **savecore**(8)

The SYNOPSIS portion of each device entry in Section 4 of this manual.

Installing SunOS 4.1
System and Network Administration

NAME

copy_home – fetch default startup files for new home directories

SYNOPSIS

/home/*groupname***/copy_home /home/***groupname* **/home/***username*

AVAILABILITY

Available only on Sun 386i systems running a SunOS 4.0.*x* release or earlier. Not a SunOS 4.1 release feature.

DESCRIPTION

Whenever **snap**(1) is used to add a new user account, the **copy_home** script in the selected primary group's home directory is executed to copy the default files to the new user's home directory, and also perform any additional custom setup.

copy_home copies default environment files, such as **.cshrc**, **.login**, and **.orgrc**, from a group's **defaults** directory to a new user's home directory. It is started by **user_agentd**(8) when **snap**(1) is used to create new home directories on a Sun386i home directory server.

Every new group created by **snap**(1) has a home directory, which can be accessed using **/home/***groupname*. **user_agentd**(8) copies the contents of the Sun386i's default group, **/home/users**, into the home directory of the new group. This includes the **Welcome.txt** file, the **copy_home** script, and the **defaults** directory . **copy_home** can be modified to customize the default setup environment for new users in the group.

SEE ALSO

snap(1), **user_agentd**(8)

Sun386i SNAP Administration
Sun386i Advanced Administration

NAME

crash – examine system images

SYNOPSIS

/etc/crash [**–d** *dump-file*] [**–n** *namelist-file*] [**–w** *output-file*]

DESCRIPTION

crash examines the memory image of a live or a crashed system kernel. It displays the values of system control structures, tables, and other pertinent information.

OPTIONS

–d *dump-file* Specify the file containing the system memory image. The default is **/dev/mem**.

–n *namelist-file*

Specify the text file containing the symbol table for symbolic access to the memory image. The default is **/vmunix**. If a system image from another machine is to be examined, the image file must be copied from that machine.

–w *output-file* Specify a file for **crash** output. The default is the standard output.

USAGE

For commands that pertain to a process, the default process is the one currently running on a live system, or the one that was running at the time the system crashed.

If the contents of a table are being dumped, the default is all active table entries.

Numeric Notation

Depending on the command, numeric arguments are assumed to be in a specific base. Counts are assumed to be decimal. Addresses are always hexadecimal. Table addresses larger than the size of the specified table are interpreted as hexadecimal addresses; smaller arguments are assumed to be in decimal. The default base of any argument may be overridden; the C conventions for designating the base of a number are recognized. (A number that is usually interpreted as decimal will be 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**.)

Expressions

Many commands accept several forms of an 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 "+" (plus sign), "–" (minus sign), "*" (multiplication symbol), "/" (division symbol), "&" (logical AND), or "|" (logical OR). An operand which is a number should be preceded by a radix prefix if it is not a decimal number (**0** for octal, **0x** for hexidecimal, **0b** for binary). The expression must be enclosed in '()' (parentheses). Other commands accept any of these argument forms that are meaningful.

Two abbreviated arguments to **crash** commands are used throughout. Both accept data entered in several forms. A *table_entry* argument may be an address, symbol, range or expression that resolves to one of these. A *start_addr* argument may be an address, symbol, or expression that resolves to one of those.

Commands

? [**–w** *filename*]

List available commands.

–w *filename*

Redirect the output of a command to the named file. Corresponds to the **redirect** command.

!*command*

Escape to the shell to execute a command.

adv [**–ep**] [**–w** *filename*] [*table_entry*] ...
Print the advertise table.

–e Display every entry in a table.

–p Interpret all address arguments in the command line as physical addresses. With this option, all address and symbol arguments explicitly entered on the command line are interpreted as physical addresses. Corresponds to the **mode** command.

as [–wfilename] [**–p**] *proc_entry* | *#pid* [**s**]]
Print the address space table.

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

buffer [**–w** *filename*] [*–format*] *bufferslot*
buffer [**–p**] [**–w** *filename*] [*–format*] *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; **–r**, directory; 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 [**–fp**] [**–w** *filename*] [*table_entry*] ...
Alias: **buf**.
Print system buffer headers.

–f Display the full structure.

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

ctx [**–w***filename*] [[**–p**] *tbl_entry* ...]
Print the context table.

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

dblock [**–ep**] [**–w** *filename*] [*dblk_addr*] ...
Print allocated streams data block headers. If the class option (**–c**) is used, only data block headers for the class specified will be printed.

defproc [**–c**] [**–w** *filename*]
defproc [**–w** *filename*] [*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.

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

file [**–ep**] [**–w** *filename*] [*table_entry*] ...
Alias: **f**.
Print the file table.

findaddr [**–w** *filename*] *table slot*
Print the address of *slot* in *table*. Only tables available to the **size** command are available to **findaddr**.

gdp [**–efp**] [**–w** *filename*] [*table_entry*] ...
Print the gift descriptor protocol table.

help [**–w** *filename*] *command* ...
Print a description of the named command, including syntax and aliases.

inode [**–f**] [**–w** *filename*] [*table_entry*] ...
Alias: **i.**
Print the inode table, including file system switch information.

kfp [**–r**] [**–s** *process*] [**–w** *filename*]
kfp [**–s** *process*] [**–w** *filename*] [*value*]
Print the frame pointer 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. If no argument is entered, the current value of the **kfp** is printed.

–s *process* Specify a process slot other than the default. Corresponds to the **defproc** command.

linkblk [**–ep**] [**–w** *filename*] [*table_entry*] ...
Print the **linkblk** table.

map [**–w** *filename*] *mapname* ...
Alias: **m.**
Print the map structure of *mapname*.

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

mblock [**–ep**] [**–w** *filename*] [*mblk_addr*] ...
Print allocated streams message block headers.

mode [**–w** *filename*] [*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 [**–p**] [**–w** *filename*] [*table_entry*] ...
Alias: **m.**
Print the mount table.

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

od [**–p**] [**–w** *filename*] [*–format*] [*–mode*] [**–s** *process*] *start_addr* [*count*]
Alias: **rd.**
Print *count* values starting at the start address in one of the following formats:

–c character
–d decimal
–x hexadecimal
–o octal
–a ASCII
–h hexadecimal character

and one of the following modes:

–l long
–t short
–b byte

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***filename*] [[**–p**] *tbl_entry*] ...
Alias: **p**.
Print the page structures.

pcb [**–w** *filename*] [*process*]
Print the process control block. If no arguments are given, the active **pcb** for the current process is printed. **–ep**

pment [**–p**] [**–w***filename*] *tbl_entry* ...
Print the page map entry table (not available on machines with a **sun3x** kernel architecture).

pmgrp [**–w***filename*] [[**–p**] *tbl_entry* ...]
Print the page map group table (not available on machines with a **sun3x** kernel architecture).

proc [**–fp**] [**–w** *filename*] [*#pid*] ... [*table_entry*] ...
proc [**–fr**] [**–w** *filename*]
Print the process table. Process table information may be specified in two ways. First, any mixture of table entries and process IDs (PID) may be entered. Each PID must be preceded by a '#' (pound sign). Alternatively, process table information for runnable processes may be specified with the runnable option (**–r**).

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

queue [**–p**] [**–w** *filename*] [*queue_addr*] ...
Print stream queues.

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

rcvd [**–efp**] [**–w** *filename*] [*table_entry*] ...
Print the receive descriptor table.

redirect [**–c**] [**–w** *filename*]
redirect [**–w** *filename*] [*filename*]
Alias: **rd**.
Used with a name, redirects output of a **crash** session to the named file. 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. To pipe output from a single **crash** command, use an exclamation point followed by a shell command:

crash-command **!** *shell-command*

This is not available when **–w** is in effect.

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

seg [**–w***filename*] [[**–p**] *proc_entry*]
seg [**–w***filename*] [*#procid* ...]
Print the segment table of process.

segdata [**–w***filename*] [[**–p**] *proc_entry*]
segdata [**–w***filename*] [*#procid* ...]
Print the segment data of process.
size [**–x**] [**–w** *filename*] [*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 [**–efp**] [**–w** *filename*] [*table_entry*] ...
Print the send descriptor table.
srmount [**–ep**] [**–w** *filename*] [*table_entry*] ...
Print the server mount table.
stack [**–u**] [**–w** *filename*] [*process*]
stack [**–k**] [**–w** *filename*] [*process*]
stack [**–p**] [**–w** *filename*] **–i** *start_addr*]
Alias: **s**.
Dump stack. The **–u** option prints the user stack. The **–k** option prints the kernel stack. The **–i** option prints the interrupt stack starting at the start address. 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.

status [**–w** *filename*]
Print system statistics.

stream [**–efp**] [**–w** *filename*] [*table_entry*] ...
Print the streams table.

strstat [**–w** *filename*]
Print streams statistics.

trace [**–r**] [**–w** *filename*] [*process*]
trace [**–p**] [**–w** *filename*] **–i** *start_addr*]
Alias: **t**.
Print stack trace. The **kfp** value is used with the **–r** option. The interrupt option prints a trace of the interrupt stack beginning at the start address. The interrupt stack trace and the stack trace for the current process are not available on a running system.

ts [**–w** *filename*] *virtual_address* ...
Print closest text symbol to the designated address.

user [**–f**] [**–w** *filename*] [*process*]
Alias: **u**.
Print the ublock for the designated process.

vfs [**–w***filename*] [[**–p**] *tbl_entry* ...]
Print the vfs table.

vnode [**–w***filename*] [[**–p**] *addr*]
Alias: **v**.
Print the vnode table.

vtop [**–s** *process*] [**–w** *filename*] *start_addr* ...
Print the physical address translation of the virtual start address.

FILES

/dev/mem	system image of currently running system
/var/crash/*machine***/vmcore.***N*	
/var/crash/*machine***/vmunix.***N*	

SEE ALSO

savecore(8)

NAME

cron – clock daemon

SYNOPSIS

/usr/etc/cron

DESCRIPTION

cron executes commands at specified dates and times. Regularly scheduled commands can be specified according to instructions found in **crontab** files in the directory **/var/spool/cron/crontabs**. Users can submit their own **crontab** files using the **crontab**(1) command. Commands that are to be executed only once may be submitted using the **at**(1) command.

cron only examines **crontab** files and **at** command files during process initialization and when a file changes using **crontab** or **at**. This reduces the overhead of checking for new or changed files at regularly scheduled intervals.

Since **cron** never exits, it should only be executed once. This is normally done by running **cron** from the initialization process through the file **/etc/rc**; see **init**(8). **/var/spool/cron/FIFO** is a FIFO file that **crontab** and **at** use to communicate with **cron**; it is also used as a lock file to prevent the execution of more than one **cron**.

FILES

/var/spool/cron	main cron directory
/var/spool/cron/FIFO	FIFO for sending messages to **cron**
/var/spool/cron/crontabs	directory containing **crontab** files

SEE ALSO

at(1), **crontab**(1), **sh**(1), **queuedefs**(5), **init**(8), **syslogd**(8)

DIAGNOSTICS

cron logs various errors to the system log daemon, **syslogd**(8), with a facility code of **cron**. The messages are listed here, grouped by severity level.

Err Severity

Can't create /var/spool/cron/FIFO: *reason*
cron was unable to start up because it could not create **/var/spool/cron/FIFO**.

Can't access /var/spool/cron/FIFO: *reason*
cron was unable to start up because it could not access **/var/spool/cron/FIFO**.

Can't open /var/spool/cron/FIFO: *reason*
cron was unable to start up because it could not open **/var/spool/cron/FIFO**.

Can't start cron - another cron may be running (/var/spool/cron/FIFO exists)
cron found that **/var/spool/cron/FIFO** already existed when it was started; this normally means that **cron** had already been started, but it may mean that an earlier **cron** terminated abnormally without removing **/var/spool/cron/FIFO**.

Can't stat /var/spool/cron/FIFO: *reason*
cron could not get the status of **/var/spool/cron/FIFO**.

Can't change directory to *directory*:*reason*
cron could not change to *directory*.

Can't read *directory*:*reason*
cron could not read *directory*.

error reading message: *reason*
An error occurred when **cron** tried to read a control message from **/var/spool/cron/FIFO**.

message received — bad format
A message was successfully read by **cron** from **/var/spool/cron/FIFO**, but the message was not of a form recognized by **cron**.

SIGTERM
received **cron** was told to terminate by having a SIGTERM signal sent to it.

cron could not unlink /var/spool/cron/FIFO: *reason*
cron was told to terminate, but it was unable to unlink **/var/spool/cron/FIFO** before it terminated.

********* CRON ABORTED **********
cron terminated, either due to an error or because it was told to.

Can't open queuedefs file *file*:*reason*
cron could not open a *queuedefs* file.

I/O error reading queuedefs file *file*:*reason*
An I/O error occurred while **cron** was reading a *queuedefs* file.

Using default queue definitions
An error occurred while trying to read a *queuedefs* file; the default queue definitions will be used.

Can't allocate *number* **bytes of space**
An internal error occurred in **cron** while trying to allocate memory.

Info Severity

queue **queue max run limit reached**
There were more jobs running or to be run in the queue *queue* than the maximum number specified. **cron** will wait until one of the currently-running jobs completes before starting to run a new one.

MAXRUN (25) procs reached
There were more than 25 jobs running or to be run by **cron**. **cron** will wait until one of the currently-running jobs completes before starting to run a new one.

***** cron started *****
cron started running.

> CMD: *pid queue command job*
A **cron** job was started, in queue *queue*, with process ID *pid*. *command* is the command to be run. For **at** or **batch** jobs, *job* is the job number.

> *user pid queue time job*
A **cron** job was started for user *user*, in queue *queue*, with process ID *pid*, at the date and time *time*. For **at** or **batch** jobs, *job* is the job number.

< *user pid queue time job status*
A **cron** job completed for user *user*, in queue *queue*, with process ID *pid*, at the date and time *time*. For **at** or **batch** jobs, *job* is the job number. If the command terminated with a non-zero exit status or a signal, *status* indicates the exit status or signal.

Notice Severity

Can't fork
An attempt to **fork** (2) to run a new job failed; **cron** will attempt again after a 30-second delay.

Warning Severity

Can't stat queuedefs file *file*:*reason*
cron could not get the status of a *queuedefs* file in order to determine whether it has changed. **cron** will assume it has changed and will reread it.

NAME

dbconfig – initializes the dial box

SYOPNSIS

/usr/etc/dbconfig *serial-device*

DESCRIPTION

dbconfig opens the designated serial port and sets its baud, parity and transmission rates. It also removes all STREAMS modules already pushed upon it (such as **ttcompat**(4M) and **ldterm**(4M)) and pushes the dial box STREAMS module "db" onto the device. **db** then holds the stream open to maintain this configuration.

If the device **/dev/dialbox** has not been created and linked to the serial port, **dbconfig** will fail.

FILES

/dev/dialbox

SEE ALSO

db(4M), **ldterm**(4M), **ttcompat**(4M), **dialtest**(6)

NAME

dcheck – file system directory consistency check

SYNOPSIS

/usr/etc/dcheck [**–i** *numbers*] [*filesystem*]

DESCRIPTION

Note: **dcheck** has been superseded for normal consistency checking by **fsck**(8).

dcheck reads the directories in a file system and compares the link-count in each inode with the number of directory entries by which it is referenced. If the file system is not specified, **dcheck** checks a set of default file systems.

dcheck is fastest if the raw version of the special file is used, since the i-list is read in large chunks.

OPTIONS

–i *numbers*

numbers is a list of i-numbers; when one of those i-numbers turns up in a directory, the number, the i-number of the directory, and the name of the entry are reported.

FILES

Default file systems vary with installation.

SEE ALSO

fs(5), **fsck**(8), **clri**(8), **icheck**(8), **ncheck**(8)

DIAGNOSTICS

When a file turns up for which the link-count and the number of directory entries disagree, the relevant facts are reported. Allocated files which have 0 link-count and no entries are also listed. The only dangerous situation occurs when there are more entries than links; if entries are removed, so the link-count drops to 0, the remaining entries point to thin air. They should be removed. When there are more links than entries, or there is an allocated file with neither links nor entries, some disk space may be lost but the situation will not degenerate.

BUGS

Since **dcheck** is inherently two-pass in nature, extraneous diagnostics may be produced if applied to active file systems.

Inode numbers less than 2 are invalid.

NAME

devinfo – print out system device information

SYNOPSIS

/usr/etc/devinfo [–v]

AVAILABILITY

This program is available on SPARCstation 1 systems only.

DESCRIPTION

devinfo displays the devices that the system knows about. The output will state the name of the device, its unit number, and whether a system device driver has claimed it. Since the internal system representation of this information is an *n*–ary tree, indentation is used to denote a parent-child relationship, and devices reported at the same indentation level are considered sibling devices.

OPTIONS

–v Report hardware specifications such as register addresses and interrupt priorities for each device.

EXAMPLE

The following example displays the format of **devinfo** output:

```
example% devinfo
Node 'Sun 4/60', unit #0 (no driver)
        Node 'options', unit #0 (no driver)
        Node 'zs', unit #0
        Node 'zs', unit #1
        Node 'fd', unit #0
        Node 'audio', unit #0
        Node 'sbus', unit #0
                Node 'dma', unit #0
                Node 'esp', unit #0
                        Node 'st', unit #1 (no driver)
                        Node 'st', unit #0
                        Node 'sd', unit #3
                        Node 'sd', unit #2
                        Node 'sd', unit #1
                        Node 'sd', unit #0
                Node 'le', unit #0
                Node 'bwtwo', unit #0
        Node 'auxiliary-io', unit #0
        Node 'interrupt-enable', unit #0
        Node 'memory-error', unit #0
        Node 'counter-timer', unit #0
        Node 'eeprom', unit #0
```

FILES

/dev/kmem to get kernel device information

NAME

devnm – device name

SYNOPSIS

/usr/etc/devnm [*name*]...

AVAILABILITY

This command is available with the *System V* software installation option. Refer to *Installing SunOS 4.1* for information on how to install optional software.

DESCRIPTION

devnm identifies the special file associated with the mounted file system where each *name* argument resides. This command can be used to construct a mount table entry for the **root** file system.

EXAMPLE

If **/usr** is mounted on **/dev/dsk/c1d0s2**, then the command:

```
/usr/etc/devnm /usr
```

produces:

```
/dev/dsk/c1d0s2 usr
```

FILES

/dev/dsk/*
/etc/mtab

SEE ALSO

fstab(5) **mount**(8)

NAME

diskusg – generate disk accounting data by user

SYNOPSIS

diskusg [**–sv**] [**–p** *filename*] [**–u** *filename*] [*filename* ...]

DESCRIPTION

diskusg generates intermediate disk accounting information from data in *filename*, or the standard input if *filename* is omitted. **diskusg** displays one line per user on the standard output in the following format:

uid login #blocks

uid is the numerical user ID of the user. *login* is the user's login name. *#blocks* is the total number of disk blocks allocated to the user.

diskusg normally reads only the i-nodes of file systems for disk accounting. In this case, *filename*s are the special filenames of these devices.

The output of **diskusg** is normally the input to **acctdisk** (see **acct**(8)) which generates total accounting records that can be merged with other accounting records. **diskusg** is normally run in **dodisk** (see **acctsh**(8)).

OPTIONS

–s	The input data is already in **diskusg** output format; combine all lines for a single user into a single line.
–v	Print a list to the standard error of all files that are not charged to any user.
–p *filename*	Use *filename* as the name of the password file to generate login names. **/etc/passwd** is used by default.
–u *filename*	Write records to *filename* of files that are not charged to any user. Records consist of the special file name, the i-node number, and the user ID.

EXAMPLES

The following example generates daily disk accounting information:

```
for i in /dev/xy0a /dev/xy0g /dev/xy1g; do
        diskusg $i > dtmp.‘basename $i‘ &
done
wait
diskusg –s dtmp.* | sort +0n +1 | acctdisk > disktacct
```

FILES

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

SEE ALSO

acct(5), **acct**(8), **acctsh**(8)

NAME

dkctl – control special disk operations

SYNOPSIS

/usr/etc/dkctl *disk command*

DESCRIPTION

dkctl is used to enable or disable special disk operations. In particular the enabling or disabling of verified writes (write check functionality) is controlled by this program.

The *disk* specification here is a disk name of the form */dev/rxxnp*, where *xx* is the controller device abbreviation (xy, sd, etc.), *n* is the disk number, and *p* is the partition to which the operation applies. The *partition* specification is simply the letter used to identify that partition in the standard UNIX system nomenclature.

SUPPORTED COMMANDS

wchk This function enables write checking for disks that support it for the named disk partition. This means that for partitions of disks with this feature enabled, all writes are *verified* to have been correctly written on the disk. This operation emphasizes data reliability over performance, although for each implementation, the fastest reasonable method will be used (i.e., implemented in hardware, if possible).

–wchk This disables write check functionality for the named disk partition.

BUGS

Use of the **dkctl** command requires super-user permissions.

There are many other features this program could control, and may in the future.

FILES

/dev/r*xxnp*

SEE ALSO

dkio(4S), **sd**(4S), **xy**(4S)

NAME

dkinfo – report information about a disk's geometry and partitioning

SYNOPSIS

/usr/etc/dkinfo *disk* [*partition*]

DESCRIPTION

dkinfo gives the total number of cylinders, heads, and sectors or tracks on the specified *disk*, and gives this information along with the starting cylinder for the specified *partition*. If no *partition* is specified on the command line, **dkinfo** reports on all partitions.

The *disk* specification here is a disk name of the form *xxn*, where *xx* is the controller device abbreviation (ip, xy, etc.) and *n* is the disk number. The *partition* specification is simply the letter used to identify that partition in the standard UNIX system nomenclature. For example, '**/usr/etc/dkinfo xy0**' reports on the first disk in a system controlled by a Xylogics controller; '**/usr/etc/dkinfo xy0g**' reports on the seventh partition of such a disk.

EXAMPLE

A request for information on my local disk, an 84 MByte disk controlled by a Xylogics 450 controller, might look like this:

```
#/usr/etc/dkinfo xy0
xy0: Xylogics 450 controller at addr ee40, unit # 0
586  cylinders 7 heads 32 sectors/track
a: 15884 sectors (70 cyls, 6 tracks, 12 sectors)
starting cylinder 0
b: 33440 sectors (149 cyls, 2 tracks)
starting cylinder 71
c: 131264 sectors (586 cyls)
starting cylinder 0
d: No such device or address
e: No such device or address
f: No such device or address
g: 81760 sectors (365 cyls)
starting cylinder 221
h: No such device or address
#
```

FILES

/dev/r*xxnp*

SEE ALSO

dkio(4S), **format**(8S)

NAME

dmesg – collect system diagnostic messages to form error log

SYNOPSIS

/usr/etc/dmesg [–]

DESCRIPTION

Note: **dmesg** is obsoleted by **syslogd**(8) for maintenance of the system error log.

dmesg looks in a system buffer for recently printed diagnostic messages and prints them on the standard output. The messages are those printed or logged by the system when errors occur. If the '–' flag is given, then **dmesg** computes (incrementally) the new messages since the last time it was run and places these on the standard output.

FILES

/var/adm/msgbuf	scratch file for memory of '–' option

SEE ALSO

syslogd(8)

NAME

dname – print RFS domain and network names

SYNOPSIS

dname [**–adn**] [**–D** *domain*] [**–N** *netspec*]

AVAILABILITY

This program is available with the *RFS* software installation option. Refer to *Installing SunOS 4.1* for information on how to install optional software.

DESCRIPTION

dname prints or defines a host's Remote File Sharing (RFS) domain name or the network used by RFS as transport provider.

When **dname** is used to change a domain name, the host's password is removed. The administrator will be prompted for a new password the next time RFS is started. See **rfstart**(8).

If **dname** is used with no options, it defaults to '**dname –d**'.

You cannot use the –D or –N options while RFS is running.

OPTIONS

–a Print both the domain name and network name.

–d Print the domain name.

–n Print the network name.

–D *domain*

Set the domain name for the host. *domain* must consist of no more than 14 characters, consisting of any combination of letters (upper and lower case), digits, hyphens (–), and underscores (_). This option is restricted to the super-user.

–N *netspec*

Set the network specification used for RFS. *netspec* is the network device name, relative to the **/dev** directory. For example, the TCP transport device, **/dev/tcp** uses **tcp**. This option is restricted to the super-user.

SEE ALSO

rfstart(8)

NOTES

This domain name is not related to the Network Interface Service (NIS) domain name. Note: NIS was formerly known as Sun Yellow Pages (YP). The functionality of the two remains the same; only the name has changed.

NAME

dorfs – initialize, start and stop RFS automatically

SYNOPSIS

dorfs init *domain netspec* [*address*]
dorfs start [–v]
dorfs stop

AVAILABILITY

This program is available with the *RFS* software installation option. Refer to *Installing SunOS 4.1* for information on how to install optional software.

DESCRIPTION

dorfs sets up necessary environment to run Remote File Sharing (RFS). You can also use it to start or stop RFS automatically, after its environment is initialized. The environment only needs to be set up once and **/usr/nserve/rfmaster** must exist before the environment is initialized. Descriptions of **/usr/nserve/rfmaster** are in **rfmaster**(5). You must be the super-user to run this command.

USAGE

Subcommands

init *domain netspec* [*address*]

domain is the name of the RFS domain. *netspec* is the name of a device file in the **/dev** directory which represents the streams-based transport provider on which RFS will run. Currently, **tcp** is the only accepted value for this field. *address* is the optional **tcp** port number on which the listener will listen. If unspecified, it defaults to **0x1450**. This subcommand only needs to be run once to initialize the environment. You do not need to rerun **dorfs** with the **init** argument, unless you want to change *netspec*. **/usr/nserve/rfmaster** must exists before you run this command to initialize the environment. To reinitialize the environment, you need to remove **/usr/nserve/***domain*, **/usr/nserve/***netspec*, **/var/net/nls/***netspec***/***address* and **/var/net/nls/***netspec***/dbf** beforehand.

start [–v]

Start RFS automatically. It also automatically advertises resources that are stored in **/etc/rstab** and mounts RFS resources that are stored in **/etc/fstab**.

–v Verify clients on mounts (see '**rfstart –v**').

stop

Takes down RFS by forced unmounting of all advertised resources, umounting all remotely mounted resources, executing **rfstop**, and stopping **listener**.

FILES

/etc/advtab
/etc/rstab
/var/net/nls/tcp/addr
/var/net/nls/tcp/dbf
/usr/nserve/domain
/usr/nserve/netspec
/usr/nserve/rfmaster

SEE ALSO

rfmaster(5), **dname**(8), **fumount**(8), **mount**(8), **nlsadmin**(8), **rfstart**(8), **rfstop**(8)

NAME

dump, rdump – incremental file system dump

SYNOPSIS

/usr/etc/dump [*options* [*arguments*]] *filesystem*
/usr/etc/dump [*options* [*arguments*]] *filename* ...

/usr/etc/rdump [*options* [*arguments*]] *filesystem*
/usr/etc/rdump [*options* [*arguments*]] *filename* ...

DESCRIPTION

dump backs up all files in *filesystem*, or files changed after a certain date, or a specified set of files and directories, to magnetic tape, diskettes, or files. *options* is a string that specifies **dump** options, as shown below. Any *arguments* supplied for specific options are given as subsequent words on the command line, in the same order as that of the *options* listed.

If **dump** is called as **rdump**, the dump device defaults to **dumphost:/dev/rmt8**.

If no *options* are given, the default is **9u**.

dump is normally used to back up a complete filesystem. To restrict the dump to a specified set of files and directories on one filesystem, list their names on the command line. In this mode the dump level is set to **0** and the **u** option is ignored.

OPTIONS

0–9 The "dump level." All files in the *filesystem* that have been modified since the last **dump** 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.

a *archive-file*
Create a dump table-of-contents archive in the specified file, *archive-file*. This file can be used by **restore**(8) to determine whether a file is present on a dump tape, and if so, on which volume it resides. For further information on the use of a dump archive file, see **restore**(8).

b *factor* Blocking factor. Specify the blocking factor for tape writes. The default is 20 blocks per write. Note: the blocking factor is specified in terms of 512 bytes blocks, for compatibility with **tar**(1). The default blocking factor for tapes of density 6250BPI and greater is 64. The default blocking factor for cartridge tapes (**c** option specified) is 126. The highest blocking factor available with most tape drives is 126.

c Cartridge. Use a cartridge instead of the standard half-inch reel. This sets the density to 1000BPI, the blocking factor to 126, and the length to 425 feet. This option also sets the "inter-record gap" to the appropriate length. When cartridge tapes are used, and this option is *not* specified, **dump** will slightly miscompute the size of the tape. If the **b**, **d**, **s** or **t** options are specified with this option, their values will override the defaults set by this option.

d *bpi* Tape density. The density of the tape, expressed in BPI, is taken from *bpi*. This is used to keep a running tab on the amount of tape used per reel. The default density is 1600 except for cartridge tape. Unless a higher density is specified explicitly, **dump** uses its default density — even if the tape drive is capable of higher-density operation (for instance, 6250BPI). Note: the density specified should correspond to the density of the tape device being used, or **dump** will not be able to handle end-of-tape properly. The **d** option is not compatible with the **D** option.

D Diskette. Specify diskette as the dump media.

f *dump-file*
Dump file. Use *dump-file* as the file to dump to, instead of **/dev/rmt8**. If *dump-file* is specified as '–', dump to the standard output. If the file name argument is of the form *machine:device*, dump to a remote machine. Since **dump** is normally run by *root*, the name of the local machine must

appear in the **.rhosts** file of the remote machine. If the file name argument is of the form *user@machine:device*, **dump** will attempt to execute as the specified user on the remote machine. The specified user must have a **.rhosts** file on the remote machine that allows root from the local machine. If **dump** is called as **rdump**, the dump device defaults to **dumphost:/dev/rmt8**. To direct the output to a desired remote machine, set up an alias for **dumphost** in the file **/etc/hosts**.

n Notify. When this option is specified, if **dump** requires attention, it sends a terminal message (similar to **wall**(1)) to all operators in the "operator" group.

s *size* Specify the *size* of the volume being dumped to. When the specified size is reached, **dump** waits for you to change the volume. **dump** interprets the specified size as the length in feet for tapes, and cartridges and as the number of 1024 byte blocks for diskettes. The following are defaults:

tape	2300 feet
cartridge	425 feet
diskette	1422 blocks (Corresponds to a 1.44 Mb diskette, with one cylinder reserved for bad block information.)

t *tracks* Specify the number of tracks for a cartridge tape. On all Sun-2 systems the default is 4 tracks, although some Sun-2 systems have 9 track drives. On all other machines the default is 9 tracks. The **t** option is not compatible with the **D** option.

u Update the dump record. Add an entry to the file **/etc/dumpdates**, for each filesystem successfully dumped that includes the filesystem name, date, and dump level. This file can be edited by the super-user.

v After writing each volume of the dump, the media is rewound and is verified against the filesystem being dumped. If any discrepancies are found, dump will respond as if a write error had occurred; the operator will be asked to mount new media, and dump will attempt to rewrite the volume. Note that *any* change to the filesystem, even the update of the access time on a file will cause the verification to fail. Thus, the verify option can only be used on a quiescent filesystem.

w List the filesystems that need backing up. This information is gleaned from the files **/etc/dumpdates** and **/etc/fstab**. When the **w** option is used, all other options are ignored. After reporting, **dump** exits immediately.

W Like **w**, but includes all filesystems that appear in **/etc/dumpdates**, along with information about their most recent dump dates and levels. Filesystems that need backing up are highlighted.

FILES

/dev/rmt8	default unit to dump to
dumphost:/dev/rmt8	default remote unit to dump to if called as **rdump**
/dev/rst*	Sun386i cartridge tape dump device
/dev/rfd0a	Sun386i 1.44 megabyte 3.5-inch high density diskette drive dump device
/dev/rfdl0a	Sun386i 720 kilobyte 3.5-inch low density diskette drive dump device
/dev/rfd0c	Sun386i 1.44 megabyte 3.5-inch high density diskette drive dump device
/dev/rfdl0c	Sun386i 720 kilobyte 3.5-inch low density diskette drive dump device
/etc/dumpdates	dump date record
/etc/fstab	dump table: file systems and frequency
/etc/group	to find group *operator*
/etc/hosts	

SEE ALSO

bar(1), **fdformat**(1), **tar**(1), **wall**(1), **dump**(5), **fstab**(5), **restore**(8), **shutdown**(8)

DIAGNOSTICS

While running, **dump** emits many verbose messages.

Exit Codes

0	Normal exit.
1	Startup errors encountered.
3	Abort – no checkpoint attempted.

BUGS

Fewer than 32 read errors on the file system are ignored.

Each reel requires a new process, so parent processes for reels already written just hang around until the entire tape is written.

It is recommended that incremental dumps also be performed with the system running in single-user mode.

dump does not support multi-file multi-volume tapes.

NOTES

Operator Intervention

dump requires operator intervention on these conditions: end of volume, end of dump, volume write error, volume open error or disk read error (if there are more than a threshold of 32). In addition to alerting all operators implied by the **n** option, **dump** interacts with the operator on **dump**'s control terminal at times when **dump** can no longer proceed, or if something is grossly wrong. All questions **dump** poses *must* be answered by typing **yes** or **no**, as appropriate.

Since backing up a disk can involve a lot of time and effort, **dump** checkpoints at the start of each volume. If writing that volume fails for some reason, **dump** will, with operator permission, restart itself from the checkpoint after a defective volume has been replaced.

dump reports periodically, and in verbose fashion. Each report includes estimates of the percentage of the dump completed and how long it will take to complete the dump. The estimated time is given as *hours*:*minutes*.

Suggested Dump Schedule

It is vital to perform full, "level **0**", dumps at regular intervals. When performing a full dump, bring the machine down to single-user mode using **shutdown**(8). While preparing for a full dump, it is a good idea to clean the tape drive and heads.

Incremental dumps allow for convenient backup and recovery on a more frequent basis of active files, with a minimum of media and time. However there are some tradeoffs. First, the interval between backups should be kept to a minimum (once a day at least). To guard against data loss as a result of a media failure (a rare, but possible occurrence), it is a good idea to capture active files on (at least) two sets of dump volumes. Another consideration is the desire to keep unnecessary duplication of files to a minimum to save both operator time and media storage. A third consideration is the ease with which a particular backed-up version of a file can be located and restored. The following four-week schedule offers a reasonable trade-off between these goals.

	Sun	*Mon*	*Tue*	*Wed*	*Thu*	*Fri*
Week 1:	**Full**	**5**	**5**	**5**	**5**	**3**
Week 2:		**5**	**5**	**5**	**5**	**3**
Week 3:		**5**	**5**	**5**	**5**	**3**
Week 4:		**5**	**5**	**5**	**5**	**3**

Although the Tuesday — Friday incrementals contain "extra copies" of files from Monday, this scheme assures that any file modified during the week can be recovered from the previous day's incremental dump.

Process Priority of dump

dump uses multiple processes to allow it to read from the disk and write to the media concurrently. Due to the way it synchronizes between these processes, any attempt to run dump with a **nice** (process priority) of '–5' or better will likely make **dump** run *slower* instead of faster.

NAME

dumpfs – dump file system information

SYNOPSIS

/usr/etc/dumpfs *device*

DESCRIPTION

dumpfs prints out the super block and cylinder group information for the file system or special device specified. The listing is very long and detailed. This command is useful mostly for finding out certain file system information such as the file system block size and minimum free space percentage.

SEE ALSO

fs(5), **fsck**(8), **newfs**(8), **tunefs**(8)

NAME

edquota – edit user quotas

SYNOPSIS

/usr/etc/edquota [**–p** *proto-user*] *usernames...*

/usr/etc/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 and an editor is then invoked on the file. 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 super-user 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**(8) for details.)

OPTIONS

–p Duplicate the quotas of the prototypical user specified for each user specified. This is the normal mechanism used to initialize quotas for groups of users.

–t Edit the soft time limits for each file system. If the time limits are zero, the default time limits in **<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/mtab	mounted file systems

SEE ALSO

quota(1), **vi**(1), **quotactl**(2), **quotacheck**(8), **quotaon**(8), **repquota**(8)

BUGS

The format of the temporary file is inscrutable.

NAME

eeprom – EEPROM display and load utility

SYNOPSIS

eeprom [–] [**–c**] [**–i**] [**–f** *device*] [*field*[=*value*] . . .]

SYNOPSIS — SPARCstation 1 SYSTEMS

eeprom [–] [**–f** *device*] [*field*[=*value*] . . .]

DESCRIPTION

eeprom displays or changes the values of fields in the EEPROM. It processes fields in the order given. When processing a *field* accompanied by a *value*, **eeprom** makes the indicated alteration to the EEPROM; otherwise it displays the *field*'s value. When given no field specifiers, **eeprom** displays the values of all EEPROM fields. A '–' flag specifies that fields and values are to be read from the standard input (one *field* or *field*=*value* per line).

Only the super-user may alter the EEPROM contents.

eeprom verifies the EEPROM checksums and complains if they are incorrect; if the **–i** flag is specified, erroneous checksums are ignored. If the **–c** flag is specified, all incorrect checksums are recomputed and corrected in the EEPROM.

The PROM monitor supports three security modes designated by the *secure* field: non-secure, command secure, and fully secure.

If *secure*=**none** the PROM monitor runs in the non-secure mode. In this mode all PROM monitor commands are allowed with no password required.

If **secure=command** the PROM monitor is in the command secure mode. In this mode, only the **b** (boot) command with no parameters and the **c** (continue) command with no parameters may be entered without a password being required. Any other command requires that the PROM monitor password be entered.

If **secure=full** the PROM monitor is in the fully secure mode. In this mode, only the **c** (continue) command with no parameters may be entered without a password being required. Entry of any other command requires that the PROM monitor password be entered. Note: the system will not auto-reboot in fully secure mode. The PROM monitor password must be entered before the boot process will take place.

When changing the security mode from non-secure to either command secure or fully secure, **eeprom** prompts for the entry and re-entry of a new PROM password as in the **passwd**(1) command. Changing from one secure mode to the other secure mode, or to the non-secure mode does not prompt for a password. Changing to non-secure mode erases the password.

The content of the **password** field is never displayed to any user. If the security mode is not **none**, the super-user may change the PROM monitor password by entering:

example# eeprom password=

eeprom prompts for a new password to be entered and re-entered.

The field **bad_login** maintains the count of bad login tries. It may be reset to zero (0) by specifying **bad_login=reset.**

OPTIONS

–c	Correct bad checksums. (Ignored on SPARCstation 1 systems.)
–i	Ignore bad checksums. (Ignored on SPARCstation 1 systems.)
–f *device*	Use *device* as the EEPROM device.

FIELDS and VALUES

hwupdate	a valid date (including **today** and **now**)
memsize	8 bit integer (megabytes of memory on machine)
memtest	8 bit integer (megabytes of memory to test)
scrsize	**1024x1024, 1152x900, 1600x1280,** or **1440x1440**

watchdog_reboot	**true** or **false**
default_boot	**true** or **false**
bootdev	*charchar*(*hex-int*,*hex-int*,*hex-int*) (with *char* a character, and *hex-int* a hexadecimal integer.)
kbdtype	8 bit integer (0 for all Sun keyboards)
keyclick	**true** or **false**
console	**b&w** or **ttya** or **ttyb** or **color**
custom_logo	**true** or **false**
banner	banner string
diagdev	%c%c (%x,%x,%x) — diagnostic boot device
diagpath	diagnostic boot path
ttya_no_rtsdtr	**true** or **false**
ttyb_no_rtsdtr	**true** or **false**
ttya_use_baud	**true** or **false**
ttyb_use_baud	**true** or **false**
ttya_baud	baud rate (16-bit decimal integer)
ttyb_baud	baud rate (16-bit decimal integer)
columns	number of columns on screen (8-bit integer)
rows	number of rows on screen (8-bit integer)
secure	**none**, **command**, or **full**
bad_login	number of bad login tries (16-bit unsigned integer, 0 if **reset**)
password	PROM monitor password (8-bytes)

FIELDS and VALUES — SPARCstation 1 SYSTEMS

hardware-revision	7 chars (for example, **30Mar88**)
selftest-#megs	32 bit decimal integer (megabytes of memory to test)
watchdog-reboot?	**true** or **false**; **true** to reboot after watchdog reset
boot-from	A string specifying boot string (for example, **le()vmunix**); defaults to **vmunix**
keyboard-click?	**true** or **false**; **true** to enable clicking of keys on each keystroke
input-device	A string specifying one of **keyboard**, **ttya**, or **ttyb**; if the specified device is unavailable, **ttya** is used for both input and output *only* if input-device specified the keyboard *and* output-device specified the screen.
output-device	A string specifying one of **screen**, **ttya**, or **ttyb**; if the specified device is unavailable, **ttya** is used for *both* input and output *only* if input-device specified the keyboard *and* output-device specified the screen.
oem-banner?	**true** or **false**; **true** to use custom banner string instead of Sun banner
oem-banner	80 chars for custom banner string
oem-logo?	**true** or **false**; **true** to display custom logo instead of Sun logo
oem-logo	Name of file (in **iconedit** format) containing custom logo.
boot-from-diag	80 chars specifying diag boot string (for example, **sd()dexec**); defaults to **le()vmunix**
ttya-mode	16 chars to specify 5 comma-separated fields of configuration information (for example, **1200,8,1,n,–**); defaults to **9600,8,1,n,–**. Fields, in left-to-right order, are: baud rate: 110, 300, 1200, 4800, 9600... data bits: 5, 6, 7, 8 parity: n(none), e(even), o(odd), m(mark), s(space) stop bits: 1, 1.5, 2 handshake: –(none), h(hardware:rts/cts), s(software:xon/xoff)
ttyb-mode	16 chars to specify 5 comma-separated fields of configuration information (for example, **1200,7,1,n,s**); defaults to **9600,8,1,n,–**.

Fields, in left-to-right order, are:

baud rate: 110, 300, 1200, 4800, 9600 ...
data bits: 5, 6, 7, 8
stop bits: 1, 1.5, 2
parity: n(none), e(even), o(odd), m(mark), s(space)
handshake: –(none), h(hardware:rts/cts), s(software:xon/xoff)

ttyb-rts-dtr-off **true** or **false**. Defaults to **false**.

ttya-rts-dtr-off **true** or **false**. Defaults to **false**.

ttya-ignore-cd **true** or **false**. Defaults to **true**.

ttyb-ignore-cd **true** or **false**; **true** to ignore the CARRIER DETECT line. Defaults to **true**.

screen-#rows number of rows on output device; defaults to 34 (for some devices actual values used may be less)

screen-#columns number of columns on output device; defaults to 80 (for some devices actual values used may be less)

auto-boot? **true** or **false**; **true** to boot on power-on

scsi-initiator-id An integer between 0 and 7 that specifies the SCSI initiator ID of the onboard SCSI host adapter.

sd-targets An array of 8 integers that map SCSI disk unit numbers to SCSI target numbers. The unit number is used to index into this string. The default settings are **31204567**, which means that unit 0 maps to target 3, unit 1 maps to target 1, and so on.

st-targets An array of 8 integers that map SCSI tape unit numbers to SCSI target numbers. The unit number is used to index into this string. The default settings are **45670123**, which means that unit 0 maps to target 4, unit 1 maps to target 5, and so on.

sunmon-compat? **true** or **false**. Defaults to **true**.

sbus-probe-list Defaults to 0123.

fcode-debug? **true** or **false**. Defaults to **false**.

last-hardware-update Date the CPU board was manufactured or upgraded to the latest hardware revision. The format is a human-readable date string, such as **23May89**.

testarea Defaults to 0.

mfg-switch? **true** or **false**. Defaults to **false**.

diag-switch? **true** or **false**. Defaults to **true**.

FILES

/dev/eeprom

FILES — SPARCstation 1 SYSTEMS

/dev/openprom

SEE ALSO

passwd(1)

PROM User's Manual

NAME

etherd, rpc.etherd – Ethernet statistics server

SYNOPSIS

/usr/etc/rpc.etherd *interface*

AVAILABILITY

This program is available with the *Networking* software installation option. Refer to *Installing SunOS 4.1* for information on how to install optional software.

DESCRIPTION

etherd is a server which puts *interface* into promiscuous mode, and keeps summary statistics of all the packets received on that interface. It responds to RPC requests for the summary. You must be root to run **etherd**.

interface is a networking interface such as **ie0**, **ie1**, **ec0**, **ec1** and **le0**.

traffic(1C) displays the information obtained from **etherd** in graphical form.

SEE ALSO

traffic(1C)

NAME

etherfind – find packets on Ethernet

SYNOPSIS

etherfind [**–d**] [**–n**] [**–p**] [**–r**] [**–t**] [**–u**] [**–v**] [**–x**] [**–c** *count*] [**–i** *interface*] [**–l** *length*] *expression*

AVAILABILITY

This program is available with the *Networking* software installation option. Refer to *Installing SunOS 4.1* for information on how to install optional software.

DESCRIPTION

etherfind prints out the information about packets on the ethernet that match the boolean *expression*. The short display, without the –v option, displays only the destination and src (with port numbers). When an Internet packet is fragmented into more than one ethernet packet, all fragments except the first are marked with an asterisk. With the –v option, the display is much more verbose, giving a trace that is suitable for analyzing many network problems. You must be root to invoke **etherfind**.

OPTIONS

–d Print the number of dropped packets. Not necessarily reliable.

–n Do not convert host addresses and port numbers to names.

–p Normally, the selected interface is put into promiscuous mode, so that **etherfind** has access to all packets on the ethernet. However, when the **–p** flag is used, the interface will not go promiscuous.

–r RPC mode: treat each packet as an RPC message, printing the program and procedure numbers. Routing packets are also more fully decoded using this option, and Network Interface Service (NIS) and NFS requests have their arguments printed.

–t Timestamps: precede each packet listing with a time value in seconds and hundredths of seconds since the first packet.

–u Make the output line buffered.

–v Verbose mode: print out some of the fields of TCP and UDP packets.

–x Dump the packet in hex, in addition to the line printed for each packet by default. Use the **–l** option to limit this printout.

–c *count*

Exit after receiving *count* packets. This is sometimes useful for dumping a sample of ethernet traffic to a file for later analysis.

–i *interface*

etherfind listens on *interface*. The program **netstat**(8C) when invoked with the **–i** flag lists all the interfaces that a machine has.

–l *length*

Use with the **–x** option to limit the number of bytes printed out.

expression

The syntax of *expression* is similar to that used by **find**(1). Here are the allowable primaries.

dst *destination*

True if the destination field of the packet is *destination*, which may be either an address or a name.

src *source*

True if the source field of the packet is *source*, which may be either an address or a name.

host *name*
True if either the source or the destination of the packet is *name*.

between *host1 host2*
True if either the source of the packet is *host1* and the destination *host2*, or the source is *host2* and the destination *host1*.

dstnet *destination*
True if the destination field of the packet has a network part of *destination*, which may be either an address or a name.

srcnet *source*
True if the source field of the packet has a network part of *source*, which may be either an address or a name.

srcport *port*
True if the packet has a source port value of *port*. This will check the source port value of either UDP or TCP packets (see **tcp**(4P)), and **udp**(4P)). The *port* can be a number or a name used in **/etc/services**.

dstport *port*
True if the packet has a destination port value of *port*. The *port* can be a number or a name.

less *length*
True if the packet has a length less than or equal to *length*.

greater *length*
True if the packet has a length greater than or equal to *length*.

–proto *protocol*
True if the packet is an IP packet (see **ip**(4P)) of protocol type *protocol*. *Protocol* can be a number or one of the names **icmp**, **udp**, **nd**, or **tcp**.

byte *byte op value*
True if byte number *byte* of the packet is in relation *op* to *value*. Legal values for *op* are **+**, **<**, **>**, **&**, and **|**. Thus **4=6** is true if the fourth byte of the packet has the value 6, and **20&0xf** is true if byte twenty has one of its four low order bits nonzero.

broadcast
True if the packet is a broadcast packet.

arp True if the packet is an ARP packet (see **arp**(4P)).

rarp True if the packet is a rarp packet.

–ip True if the packet is an IP packet.

–decnet
True if the packet is a DECNET packet.

–apple True if the packet is an AppleTalk protocol packet.

The primaries may be combined using the following operators (in order of decreasing precedence):

A parenthesized group of primaries and operators (parentheses are special to the Shell and must be escaped).

The negation of a primary ('**not**' is the unary *not* operator).

Concatenation of primaries (the *and* operation is implied by the juxtaposition of two primaries, or can be specified with **'and'**).

Alternation of primaries (**'or'** is the *or* operator).

EXAMPLE

To find all packets arriving at or departing from the host **sundown**, or that are ICMP packets:

example% etherfind host sundown or proto icmp

SEE ALSO

find(1), **traffic**(1C), **arp**(4P), **ip**(4P), **nit**(4P) **tcp**(4P), **udp**(4P), **netstat**(8C)

BUGS

The syntax is painful.

NOTES

The Network Information Service (NIS) was formerly known as Sun Yellow Pages (YP). The functionality of the two remains the same; only the name has changed.

NAME

exportfs – export and unexport directories to NFS clients

SYNOPSIS

/usr/etc/exportfs [**–aiuv**] [**–o** *options*] [*pathname*]

DESCRIPTION

exportfs makes a local directory or filename available for mounting over the network by NFS clients. It is normally invoked at boot time by the **/etc/rc.local** script, and uses information contained in the **/etc/exports** file to export *pathname* (which must be specified as a full pathname). The super-user can run **exportfs** at any time to alter the list or characteristics of exported directories and filenames. Directories and files that are currently exported are listed in the file **/etc/xtab**.

With no options or arguments, **exportfs** prints out the list of directories and filenames currently exported.

OPTIONS

–a All. Export all pathnames listed in **/etc/exports**, or if **–u** is specified, unexport all of the currently exported pathnames.

–i Ignore the options in **/etc/exports**. Normally, **exportfs** will consult **/etc/exports** for the options associated with the exported pathname.

–u Unexport the indicated pathnames.

–v Verbose. Print each directory or filename as it is exported or unexported.

–o *options*

Specify a comma-separated list of optional characteristics for the pathname being exported. *options* can be selected from among:

ro Export the pathname read-only. If not specified, the pathname is exported read-write.

rw=*hostname*[**:***hostname*] . . .

Export the pathname read-mostly. Read-mostly means exported read-only to most machines, but read-write to those specified. If not specified, the pathname is exported read-write to all.

anon=*uid*

If a request comes from an unknown user, use UID as the effective user ID. Note: root users (UID 0) are always considered "unknown" by the NFS server, unless they are included in the **root** option below. The default value for this option is –2. Setting the value of "anon" to –1 disables anonymous access. Note: by default secure NFS accepts insecure requests as anonymous, and those wishing for extra security can disable this feature by setting "anon" to –1.

root=*hostname*[**:***hostname*] . . .

Give root access only to the root users from a specified *hostname*. The default is for no hosts to be granted root access.

access=*client*[**:***client*] . . .

Give mount access to each *client* listed. A *client* can either be a hostname, or a netgroup (see **netgroup**(5)). Each *client* in the list is first checked for in the **/etc/netgroup** database, and then the **/etc/hosts** database. The default value allows any machine to mount the given directory.

secure Require clients to use a more secure protocol when accessing the directory.

FILES

/etc/exports	static export information
/etc/xtab	current state of exported pathnames
/etc/netgroup	

SEE ALSO

exports(5), **netgroup**(5), **showmount**(8)

WARNINGS

You cannot export a directory that is either a parent- or a sub-directory of one that is currently exported and *within the same filesystem*. It would be illegal, for example, to export both **/usr** and **/usr/local** if both directories resided in the same disk partition.

NAME

extract_patch – extract and execute patch files from installation tapes

SYNOPSIS

extract_patch [**–d***device* [**–r***remote-host*]] [**–p***patch-name*] [**–DEFAULT**]

DESCRIPTION

extract_patch extracts a patch from a release tape onto the current system. If no options are specifed, it prompts for input as to the patch name, tape device, or remote hostname from which to the software is to be installed. If the named patch cannot be found, a list of valid patches are printed.

If the named patch is found then the patch is extracted from the tape onto the system. If there is a **README** file in the extracted contents then the user is given a chance to view it. If there is a patch installation program the user is given a chance to run it.

Patches must appear in the tape's table of contents, and must have a name that starts with "Patch_".

OPTIONS

–d*device*

Install from the indicated tape drive, such as **st0**, or **mt0**.

–r*remote-host*

Install from the device given in the **–d** option on the indicated remote host.

–p*patch-name*

Specifes the name of the patch to extract.

–DEFAULT

Execute the installation script using all default values. Otherwise the installation script prompts for any optional values.

SEE ALSO

extract_unbundled(8)

NAME

extract_unbundled – extract and execute unbundled-product installation scripts

SYNOPSIS

extract_unbundled [–d*device* [–r*remote-host*]] [–DEFAULT]

DESCRIPTION

extract_unbundled extracts and executes the installation scripts from release tapes for Sun unbundled software products. If no options are specified, it prompts for input as to the tape device, or remote host-name from which to the software is to be installed. For information about installing a specific product, refer to the installation manual that accompanies that product.

OPTIONS

–d*device*

Install from the indicated tape drive, such as **st0** or **mt0**.

–r*remote_host*

Install from the device given in the **–d** option on the indicated remote host.

–DEFAULT

Execute the installation script using all default values. Otherwise the installation script prompts for any optional values.

NAME

fastboot, fasthalt – reboot/halt the system without checking the disks

SYNOPSIS

/usr/etc/fastboot [*boot-options*]

/usr/etc/fasthalt [*halt-options*]

DESCRIPTION

fastboot and **fasthalt** are shell scripts that reboot and halt the system without checking the file systems. This is done by creating a file **/fastboot**, then invoking the **reboot**(8) program. The system startup script, **/etc/rc**, looks for this file and, if present, skips the normal invocation of **fsck**(8).

FILES

/usr/etc/fastboot
/etc/rc

SEE ALSO

fsck(8), **halt**(8), **init**(8), **rc**(8), **reboot**(8)

NAME

fingerd, in.fingerd – remote user information server

SYNOPSIS

/usr/etc/in.fingerd

DESCRIPTION

fingerd implements the server side of the Name/Finger protocol, specified in RFC 742. The Name/Finger protocol provides a remote interface to programs which display information on system status and individual users. The protocol imposes little structure on the format of the exchange between client and server. The client provides a single "command line" to the finger server which returns a printable reply.

fingerd waits for connections on TCP port 79. Once connected it reads a single command line terminated by a LINEFEED which is passed to **finger**(1). **fingerd** closes its connections as soon as the output is finished.

If the line is null (only a LINEFEED is sent) then **finger** returns a "default" report that lists all people logged into the system at that moment.

If a user name is specified (for instance, ericLINEFEED) then the response lists more extended information for only that particular user, whether logged in or not. Allowable "names" in the command line include both "login names" and "user names". If a name is ambiguous, all possible derivations are returned.

SEE ALSO

finger(1)

Harrenstien, Ken, *NAME/FINGER*, RFC 742, Network Information Center, SRI International, Menlo Park, Calif., December 1977.

BUGS

Connecting directly to the server from a TIP or an equally narrow-minded TELNET-protocol user program can result in meaningless attempts at option negotiation being sent to the server, which will foul up the command line interpretation. **fingerd** should be taught to filter out IAC's and perhaps even respond negatively (IAC *will not)* to all option commands received.

NAME

fonftlip – create Sun386i-style vfont file

SYNOPSIS

fontflip fontname [**–o** *newfontname*]

AVAILABILITY

Available only on Sun 386i systems running a SunOS 4.0.*x* release or earlier. Not a SunOS 4.1 release feature.

DESCRIPTION

fontflip takes as input a vfont file (Sun-3 fixedwidthfont) and creates a Sun386i system vfont. This new font is a bitflipped version of its input. The new font is named *oldfont*.**flip** unless otherwise specified.

OPTIONS

–o *newfontname* Specify the name of the new flipped font.

FILES

/usr/lib/fonts/fixedwidthfonts

SEE ALSO

vfont(5)

NAME

format – disk partitioning and maintenance utility

SYNOPSIS

format [**–f** *command-file*] [**–l** *log-file*] [**–x** *data-file*] [**–d** *disk-name*] [**–t** *disk_type*] [**–p** *partition-name*] [**–s**] *diskname*...

DESCRIPTION

format enables you to format, label, repair and analyze disks on your Sun computer. Unlike previous disk maintenance programs, **format** runs under SunOS. Because there are limitations to what can be done to the system disk while the system is running, **format** is also supported within the memory-resident system environment. For most applications, however, running **format** under SunOS is the more convenient approach.

If no *disk-list* is present, **format** uses the disk list defined in the data file specified with the **–x** option. If that option is omitted, the data file defaults to **format.dat** in the current directory, or else **/etc/format.dat**.

OPTIONS

–f *command-file*

Take command input from *command-file* rather than the standard input. The file must contain commands that appear just as they would if they had been entered from the keyboard. With this option, **format** does not issue **continue?** prompts.

–l *log-file*

Log a transcript of the **format** session to the indicated *log-file*, including the standard input, the standard output and the standard error.

–x *data-file*

Use the disk list contained in *data-file*.

–d *disk_name*

Specify which disk should be made current upon entry into the program. The disk is specified by its logical name (for instance, - xy0). This can also be accomplished by specifying a single disk in the disk list.

–t *disk-type*

Specify the type of disk which is current upon entry into the program, A disk's type is specified by name in the data file. This option can only be used if a disk is being made current as described above.

–p *partition-name*

Specify the partition table for the disk which is current upon entry into the program. The table is specified by its name as defined in the data file. This option can only be used if a disk is being made current, and its type is either specified or available from the disk label.

–s Silent. Suppress all of the standard output. Error messages are still displayed. This is generally used in conjunction with the **–f** option.

FILES

/etc/format.dat default data file

SEE ALSO

System and Network Administration

NAME

fpa_download – download to the Floating Point Accelerator

SYNOPSIS

fpa_download [**–d**] [**–r**] [**–v**] [**–u** *ufile*] [**–m** *mfile*] [**–c** *cfile*]

AVAILABILITY

fpa_download applies to Sun-3 and Sun-3x systems equipped with either an FPA or FPA+.

DESCRIPTION

fpa_download writes microcode, map, and constants files to FPA and FPA+ boards. FPA requires a map file; FPA+ does not.

Root execution level is required to download (d,u,m and c options). **fpa_download** is called from **/etc/rc.local** when **/dev/fpa** exists.

Given no arguments, **fpa_download** prints whether an FPA, or FPA+ is installed.

OPTIONS

–d	Download microcode, constants, and map files. Enable default file names.
–r	Print microcode and constant revision.
–v	Verbose mode.
–u *ufile*	Download microcode from *ufile*.
–m *mfile*	Download map from *mfile* (FPA only).
–c *cfile*	Download constants from *cfile*.

FILES

/dev/fpa	device file for both FPA and FPA+.
/usr/etc/fpa/fpa_micro_bin	default microcode file (*ufile*) for FPA.
/usr/etc/fpa/fpa_constants	default constants file (cfile) for FPA
/usr/etc/fpa/fpa_micro_map	default map file (mfile) for FPA
/usr/etc/fpa/fpa_micro_bin+	default microcode file (ufile) for FPA+
/usr/etc/fpa/fpa_constants+	default constants file (cfile) for FPA+

SEE ALSO

fpa(4)

DIAGNOSTICS

The following diagnostics are printed when **fpa_download** encounters a serious error and asks the kernel to disable the FPA. This might occur if the microcode, map, or constants files are corrupted, or if there is an FPA or system hardware problem.

FPA Download Failed - FPA ioctl failed
An ioctl() on **/dev/fpa** failed, possibly due to a hung FPA pipe.

FPA Failed Download - FPA Bus Error
Received a SIGFPE.

FPA Failed Download - Upload mismatch
After each file is written to the FPA/FPA+, **fpa_download** uploads the contents of FPA memory and compares it with the source. They should always match.

NAME

fparel – Sun FPA online reliability tests

SYNOPSIS

fparel [**–p***n*] [**–v**]

AVAILABILITY

Not available on Sun386i systems.

DESCRIPTION

fparel is a command to execute the Sun FPA online confidence and reliability test program. **fparel** tests about 90% of the functions of the FPA board, and tests all FPA contexts not in use by other processes. **fparel** runs without disturbing other processes that may be using the FPA. **fparel** can only be run by the super-user.

After a successful pass, **fparel** writes

time, date: Sun FPA Passed. The contexts tested are: 0, 1, ... 31

to the file **/var/adm/diaglog.**

If a pass fails, **fparel** writes

time, date: Sun FPA failed

along with the test name and context number that failed, to the file **/var/adm/diaglog. fparel** then broadcasts the message

time, date: Sun FPA failed, disabled, service required

to all users of the system. Next, **fparel** causes the kernel to disable the FPA. Once the kernel disables the FPA, the system must be rebooted to make it accessible.

The file **/etc/rc.local** should contain an entry to cause **fparel** to be invoked upon reboot to be sure that the FPA remains unaccessible in cases where rebooting doesn't correct the problem. See **rc**(8).

The **crontab**(5) file for root should contain an entry indicating that **cron**(8) is to run **fparel** daily, such as:

7 2 * * * /usr/etc/fpa/fparel

which causes **fparel** to run at seven minutes past two, every day. See **cron**(8) and **crontab**(5) for details.

OPTIONS

–p*n* Perform *n* passes. Default is *n*=1. **–p0** means perform 2147483647 passes.

–v Run in verbose mode with detailed test results to the standard output.

FILES

/var/adm/diaglog Log of **fparel** diagnostics.
/etc/rc.local
/var/spool/cron/crontabs/root
/usr/etc/fpa/* directory containing FPA microcode, data files, and loader

SEE ALSO

crontab(5), **cron**(8), **fpaversion**(8), **rc**(8)

NAME

fpaversion – print FPA version, load microcode

SYNOPSIS

fpaversion [–chlqv] [–t [cdhimprstvxCIMS]]

AVAILABILITY

Available only on Sun-3 and Sun-3x systems equipped with either an FPA or an FPA+.

DESCRIPTION

fpaversion performs various tests on the FPA or FPA+. Without arguments, it prints the microcode version number and constants currently installed on **/dev/fpa**. **fpaversion** also performs a quick test to ensure proper operation and reports whether an FPA or an FPA+ is installed.

OPTIONS

–c Continue tests after an error.

–h Help. Print command-line summary.

–l Loop through tests infinitely.

–q Quiet output. Print out only error messages.

–v Verbose output.

–t Specify certain tests:

- **c** Command register format instructions.
- **d** Double precision format instructions.
- **h** Help. Print summary of test specifiers.
- **i** Imask register.
- **m** Mode register.
- **p** Simple pipe sequencing.
- **r** User registers for all contexts.
- **s** Single precision format instructions.
- **t** Status generation.
- **v** Print version number and date of microcode, and constants. Report whether an FPA or an FPA+ is installed.
- **x** Extended format instructions.
- **C** Check checksum for microcode, mapping RAM, and constant RAM for the FPA. Check checksum for microcode RAM and constant RAM for the FPA+.
- **I** Allows interactive reads and writes to the FPA.
- **M** Command register format matrix instructions.
- **S** Shadow registers.

FILES

/dev/fpa	physical FPA device
/usr/etc/fpa/fpa_micro_bin	microcode binaries for the FPA
/usr/etc/fpa/fpa_micro_map	microcode map binaries for the FPA
/usr/etc/fpa/fpa_constants	microcode data file for the FPA
/usr/etc/fpa/fpa_micro_bin+	microcode binaries for the FPA+
/usr/etc/fpa/fpa_constants+	microcode data file for the FPA+
/usr/etc/fpa/fpa_download	microcode loader

SEE ALSO

fpa_download(8), **fparel**(8), **sundiag**(8)

DIAGNOSTICS

If a test fails, its name, along with the actual and expected results will be printed.

NAME

fpurel – perform tests the Sun Floating Point Co-processor.

SYNOPSIS

fpurel [**–v**] [**–p**[*count*]] [**–r**]

DESCRIPTION

fpurel performs a series of functional and computational tests for the Sun Floating Point Co-processor to verify that it is operational and accurate. With no options, **fpurel** runs one pass silently in the foreground and only reports errors if any are found.

OPTIONS

–v Verbose. Display the name and results of each test on the console. The default is to run silently.

–p[*count*]
Passcount. Specify the number of times to run the test suite. The default is to run one pass.

–r Disable stop on error. Continue to run if errors are detected. The default is to display the error message and to stop testing when an error is detected.

EXAMPLE

This example uses **fpurel** from the **/usr/diag** directory. If no errors are detected, then no information is displayed.

% /usr/diag/fpurel

NAME

fpuversion4 – print the Sun-4 FPU version

SYNOPSIS

/usr/etc/fpuversion4

AVAILABILITY

Sun-4 systems only.

DESCRIPTION

fpuversion4 reads the **%fsr** register to determine the FPU version installed on a Sun-4. The printed version field contains a value in the range 0-7; by SPARC convention 7 indicates that no FPU is installed, so floating-point instructions are always emulated in the kernel.

NAME

fsck – file system consistency check and interactive repair

SYNOPSIS

/usr/etc/fsck **–p** [*filesystem* ...]

/usr/etc/fsck [**–b** *block#*] [**–w**] [**–y**] [**–n**] [**–c**] [*filesystem*] ...

DESCRIPTION

The first form of **fsck** preens a standard set of file systems or the specified file systems. It is normally used in the **/etc/rc** script during automatic reboot. In this case, **fsck** reads the table **/etc/fstab** to determine the file systems to check. It inspects disks in parallel, taking maximum advantage of I/O overlap to check the file systems as quickly as possible.

Normally, the root file system is checked in pass 1; other root-partition file systems are checked in pass 2. Small file systems on separate partitions are checked in pass 3, while larger ones are checked in passes 4 and 5.

Only partitions marked in **/etc/fstab** with a file system type of "4.2" and a non-zero pass number are checked.

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.

If **fsck** encounters other inconsistencies that it cannot fix automatically, it exits with an abnormal return status (and the reboot fails).

If sent a QUIT signal, **fsck** will finish the file system checks, then exit with an abnormal return status that causes the automatic reboot to fail. This is useful when you wish to finish the file system checks, but do not want the machine to come up multiuser.

Without the **–p** option, **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 above can often result in some loss of data. The amount and severity of data lost can be determined from the diagnostic output.

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.

If no file systems are given to **fsck** then a default list of file systems is read from the file **/etc/fstab**.

Inconsistencies checked in order 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.
- 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. 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 cooked or raw device on which it resides, or by giving the name of its mount point. If the latter is given, **fsck** finds the name of the device on which the file system resides by looking in **/etc/fstab**.

Checking the raw device is almost always faster.

OPTIONS

–b Use the block specified immediately after the flag as the super block for the file system. Block 32 is always an alternate super block.

–w Check writable file systems only.

–y Assume a **yes** response to all questions asked by **fsck**; this should be used with extreme caution, as it is a free license to continue, even after severe problems are encountered.

–n Assume a **no** response to all questions asked by **fsck**; do not open the file system for writing.

–c If the file system is in the old (static table) format, convert it to the new (dynamic table) format. If the file system is in the new format, convert it to the old format provided the old format can support the filesystem configuration. In interactive mode, **fsck** will list the direction the conversion is to be made and ask whether the conversion should be done. If a negative answer is given, no further operations are done on the filesystem. In preen mode, the direction of the conversion is listed and done if possible without user interaction. Conversion in preen mode is best used when all the file systems are being converted at once. The format of a file system can be determined from the first line of output from **dumpfs**(8)

FILES

/etc/fstab default list of file systems to check

DIAGNOSTICS

The diagnostics produced by **fsck** are fully enumerated and explained in *System and Network Administration.*

EXIT STATUS

0 Either no errors detected or all errors were corrected.

4 Root file system errors were corrected. The system must be rebooted.

8 Some uncorrected errors exist on one or more of the file systems checked, there was a syntax error, or some other operational error occurred.

12 A signal was caught during processing.

SEE ALSO

fs(5), **fstab**(5), **dumpfs**(8), **newfs**(8), **mkfs**(8), **panic**(8S), **reboot**(8), **rexecd**(8C), **ypserv**(8)

System and Network Administration

BUGS

There should be some way to start a '**fsck –p**' at pass *n*.

NAME

fsirand – install random inode generation numbers

SYNOPSIS

fsirand [**–p**] *special*

DESCRIPTION

fsirand installs random inode generation numbers on all the inodes on device *special*, and also installs a filesystem ID in the superblock. This helps increase the security of filesystems exported by NFS.

fsirand must be used only on an unmounted filesystem that has been checked with **fsck**(8). The only exception is that it can be used on the root filesystem in single-user mode, if the system is immediately rebooted afterwords.

OPTIONS

–p Print out the generation numbers for all the inodes, but do not change the generation numbers.

SEE ALSO

fsck(8)

NAME

ftpd, in.ftpd – TCP/IP Internet File Transfer Protocol server

SYNOPSIS

/usr/etc/in.ftpd [**–dl**] [**–t***timeout*] *host.socket*

AVAILABILITY

This program is available with the *Networking* software installation option. Refer to *Installing SunOS 4.1* for information on how to install optional software.

DESCRIPTION

ftpd is the TCP/IP Internet File Transfer Protocol (FTP) server process. The server is invoked by the Internet daemon **inetd**(8C) each time a connection to the FTP service (see **services**(5)) is made, with the connection available as descriptor 0 and the host and socket the connection originated from (in hex and decimal respectively) as argument.

Inactive connections are timed out after 60 seconds.

If the **–d** option is specified, debugging information is logged to the system log daemon, **syslogd**(8).

If the **–l** option is specified, each FTP session is logged to **syslogd**.

The FTP server will timeout an inactive session after 15 minutes. If the **–t** option is specified, the inactivity timeout period will be set to *timeout*.

The FTP server currently supports the following FTP requests; case is not distinguished.

Request	Description
ABOR	abort previous command
ACCT	specify account (ignored)
ALLO	allocate storage (vacuously)
APPE	append to a file
CDUP	change to parent of current working directory
CWD	change working directory
DELE	delete a file
HELP	give help information
LIST	give list files in a directory (**ls –lg**)
MKD	make a directory
MODE	specify data transfer *mode*
NLST	give name list of files in directory (**ls**)
NOOP	do nothing
PASS	specify password
PASV	prepare for server-to-server transfer
PORT	specify data connection port
PWD	print the current working directory
QUIT	terminate session
RETR	retrieve a file
RMD	remove a directory
RNFR	specify rename-from file name
RNTO	specify rename-to file name

STOR store a file

STOU store a file with a unique name

STRU specify data transfer *structure*

TYPE specify data transfer *type*

USER specify user name

XCUP change to parent of current working directory

XCWD change working directory

XMKD make a directory

XPWD print the current working directory

XRMD remove a directory

The remaining FTP requests specified in RFC 959 are recognized, but not implemented.

The FTP server will abort an active file transfer only when the **ABOR** command is preceded by a Telnet "Interrupt Process" (IP) signal and a Telnet "Synch" signal in the command Telnet stream, as described in RFC 959.

ftpd interprets file names according to the "globbing" conventions used by **csh**(1). This allows users to utilize the metacharacters '* ? [] {}~'.

ftpd authenticates users according to three rules.

- The user name must be in the password data base, **/etc/passwd**, and not have a null password. In this case a password must be provided by the client before any file operations may be performed.
- If the file **/etc/ftpusers** exists, the user name must not appear in that file.
- The user must have a standard shell returned by **getusershell**(3).
- If the user name is "anonymous" or "ftp", an anonymous FTP account must be present in the password file (user "ftp"). In this case the user is allowed to log in by specifying any password (by convention this is given as the client host's name).

In the last case, **ftpd** takes special measures to restrict the client's access privileges. The server performs a **chroot**(2) command to the home directory of the "ftp" user. In order that system security is not breached, it is recommended that the "ftp" subtree be constructed with care; the following rules are recommended.

~ftp Make the home directory owned by "ftp" and unwritable by anyone.

~ftp/bin Make this directory owned by the super-user and unwritable by anyone. The program **ls**(1V) must be present to support the list commands. This program should have mode 111. Since the default /bin/ls command is linked with a shared library, so you need to set up the files for dynamic linking as well.

~ftp/usr/lib/ld.so
the runtime loader must be present and executable.

~ftp/dev/zero
used by the runtime loader, create this with the command "mknod zero c 3 12".

~ftp/usr/lib/libc.so.*
should be a copy of the latest version of the shared C library.

~ftp/etc Make this directory owned by the super-user and unwritable by anyone. The files **passwd**(5) and **group**(5) must be present for the **ls** command to work properly. These files should be mode 444.

~ftp/pub Make this directory mode 777 and owned by "ftp". Users should then place files which are to be accessible via the anonymous account in this directory.

DIAGNOSTICS

ftpd logs various errors to the system log daemon, **syslogd**, with a facility code of **daemon**. The messages are listed here, grouped by severity level.

Err Severity

getpeername failed: *reason*
A **getpeername**(2) call failed.

getsockname failed: *reason*
A **getsockname**(2) call failed.

signal failed: *reason*
A **signal** (3V) (see **signal**(3V)) call failed.

setsockopt failed: *reason*
A **setsockopt** call (see **getsockopt**(2)) failed.

ioctl failed: *reason*
A **ioctl**(2) call failed.

directory: *reason*
ftpd did not have write permission on the directory *directory* in which a file was to be created by the STOU command.

Info Severity

These messages are logged only if the –**l** flag is specified.

FTPD: connection from *host* **at** *time*
A connection was made to **ftpd** from the host *host* at the date and time *time*.

FTPD: User *user* **timed out after** *timeout* **seconds at** *time*
The user *user* was logged out because they hadn't entered any commands after *timeout* seconds; the logout occurred at the date and time *time*.

Debug Severity

These messages are logged only if the –**d** flag is specified.

TPD: command: *command*
A command line containing *command* was read from the FTP client.

lost connection
The FTP client dropped the connection.

<--- *replycode*
<--- *replycode*–
A reply was sent to the FTP client with the reply code *replycode*. The next message logged will include the message associated with the reply. If a – follows the reply code, the reply is continued on later lines.

SEE ALSO

csh(1), **ftp**(1C), **ls**(1V), **chroot**(2) **getpeername**(2), **getsockname**(2), **getsockopt**(2), **ioctl**(2), **getusershell**(3), **ftpusers**(5), **group**(5), **passwd**(5), **services**(5), **inetd**(8C), **syslogd**(8)

Postel, Jon, and Joyce Reynolds, *File Transfer Protocol (FTP)*, RFC 959, Network Information Center, SRI International, Menlo Park, Calif., October 1985.

BUGS

The anonymous account is inherently dangerous and should be avoided when possible.

The server must run as the super-user to create sockets with privileged port numbers. It maintains an effective user ID of the logged in user, reverting to the super-user only when binding addresses to sockets. The possible security holes have been extensively scrutinized, but are possibly incomplete.

NAME

fumount – force unmount of an advertised RFS resource

SYNOPSIS

fumount [–**w** *seconds*] *resource*

AVAILABILITY

This program is available with the *RFS* software installation option. Refer to *Installing SunOS 4.1* for information on how to install optional software.

DESCRIPTION

fumount unadvertises *resource* and disconnects remote access to the resource.

When the forced unmount occurs, an administrative shell script, **rfuadmin**, is started on each remote system that has the resource mounted If a grace period is specified (in seconds), **rfuadmin**(8) is started with the **fuwarn** option. When the actual forced unmount is ready to occur, **rfuadmin**(8) is started with the **fumount** option. See **rfuadmin**(8) for information on the action taken in response to the forced unmount.

This command is restricted to the super-user.

An error message will be sent to standard error if any of the following are true of *resource*:

- It does not physically reside on the local machine.
- It is an invalid resource name.
- It is not currently advertised and is not remotely mounted.

OPTION

–**w** *seconds* Delay execution of the disconnect *seconds* seconds.

SEE ALSO

adv(8), **mount**(8), **rfuadmin**(8), **rfudaemon**(8), **unadv**(8)

NAME

fusage – RFS disk access profiler

SYNOPSIS

fusage [[*mount_point*] | [*advertised_resource*] | [*block_special_device*] [. . .]]

AVAILABILITY

This program is available with the *RFS* software installation option. Refer to *Installing SunOS 4.1* for information on how to install optional software.

DESCRIPTION

When used with no options, **fusage** reports block I/O transfers, in kilobytes, to and from all locally mounted file systems and advertised Remote File Sharing resources on a per client basis. The count data are cumulative since the time of the mount. When used with an option, **fusage** reports on the named file system, advertised resource, or block special device.

The report includes one section for each file system and advertised resource and has one entry for each machine that has the directory remotely mounted, ordered by decreasing usage. Sections are ordered by device name; advertised resources that are not complete file systems will immediately follow the sections for the file systems they are in.

SEE ALSO

df(1V), **adv**(8), **crash**(8), **mount**(8)

NAME

fuser – identify processes using a file or file structure

SYNOPSIS

/usr/etc/fuser [**–ku**] *filename* | *resource* [–] [[**–ku**] *filename* | *resource*]

DESCRIPTION

fuser outputs the process IDs of the processes that are using the *filename*s or remote *resources* specified as arguments. Each process ID is followed by a letter code. Possible code letters and an explanation of how the process is using the file are given below:

- **c** its current directory
- **p** the parent of its current directory (only when the file is being used by the system)
- **r** its root directory
- **v** process has **exec**'ed or **mmap**'ed file

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 are reported. **fuser** cannot use the mount point of the remote resource to report all processes using any file associated with that remote resource; it must use the resource name. For all other types of files (text files, executables, directories, devices, etc.) only the processes using that file are reported.

The process IDs are printed as a single line on the standard output, separated by SPACE characters and terminated with a single NEWLINE. All other output is written on standard error.

Any user with permission to read **/dev/kmem** and **/dev/mem** can use **fuser**.

Only the super-user can terminate another user's process

OPTIONS

If more than one group of files are specified, the options may be respecified for each additional group of files.

- **–** Cancel the options currently in force. The new set of options applies to the next group of files.
- **–k** Send SIGKILL signal to each process. Since this option spawns kills for each process, the kill messages may not show up immediately (see **kill**(2V)).
- **–u** User login name, in parentheses, also follows the process ID.

FILES

/vmunix	system namelist
/dev/kmem	system image
/dev/mem	system image

SEE ALSO

ps(1), **kill**(2V), **signal**(3V), **mount**(8)

NAME

fwtmp, wtmpfix – manipulate connect accounting records

SYNOPSIS

/usr/lib/acct/fwtmp [**–ci**]

/usr/lib/acct/wtmpfix [*filename* ...]

DESCRIPTION

fwtmp

fwtmp reads from the standard input and writes to the standard output, converting binary records of the type found in **wtmp** to formatted ASCII records. The ASCII version is useful to enable editing bad records, using a text editor, or general purpose maintenance of the file.

wtmpfix

wtmpfix examines the standard input or named files in **wtmp** format, corrects the time/date stamps to make the entries consistent, and writes to the standard output. A '–' can be used in place of *filename* to indicate the standard input. If time/date corrections are not performed, **acctcon1** fails 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 '|' placed in the line field of the **<utmp.h>** structure. The second record specifies the new date and is denoted by the string '{' placed in the line field. **wtmpfix** uses these records to synchronize all time stamps in the file.

In addition to correcting time/date stamps, **wtmpfix** checks the validity of the name field to ensure that it consists solely of alphanumeric characters or SPACE characters. If it encounters a name that is considered invalid, it changes the login name to **INVALID** and writes a diagnostic message to the standard error. In this way, **wtmpfix** reduces the chance that **acctcon1** will fail when processing connect accounting records.

OPTIONS

fwtmp

–c Write output in binary form.

–i Input is in ASCII form.

FILES

/var/adm/wtmp

SEE ALSO

acctcom(1), **acct**(2V), **acct**(5), **utmp**(5V), **acct**(8), **acctcms**(8), **acctcon**(8), **acctmerg**(8), **acctprc**(8), **acctsh**(8), **runacct**(8)

NAME

gettable – get DARPA Internet format host table from a host

SYNOPSIS

/usr/etc/gettable *host*

DESCRIPTION

gettable is a simple program used to obtain the DARPA Internet host table from a "hostname" server. The indicated *host* is queried for the table. The table, if retrieved, is placed in the file **hosts.txt**.

gettable operates by opening a TCP connection to the port indicated in the service specification for "hostname" . A request is then made for "ALL" names and the resultant information is placed in the output file.

gettable is best used in conjunction with the **htable**(8) program which converts the DARPA Internet host table format to that used by the network library lookup routines.

SEE ALSO

intro(3), **htable**(8)

Harrenstien, Ken, Mary Stahl, and Elizabeth Feinler, *HOSTNAME Server*, RFC 953, Network Information Center, SRI International, Menlo Park, Calif., October 1985.

BUGS

Should allow requests for only part of the database.

NAME

getty – set terminal mode

SYNOPSIS

/usr/etc/getty [*type* [*tty*]]

Sun386i SYSTEM SYNOPSIS

/usr/etc/getty [**–n**] [*type* [*tty*]]

DESCRIPTION

getty, which is invoked by **init**(8), opens and initializes a tty line, reads a login name, and invokes **login**(1).

The *tty* argument is the name of the character-special file in **/dev** that corresponds to the terminal. If there is no *tty* argument, or the argument is '–', the tty line is assumed to be opened as file descriptor **0**.

The *type* argument, if supplied, is used as an index into the **gettytab**(5) database—to determine the characteristics of the line. If this argument is absent, or if there is no such entry, the default entry is used. If there is no **/etc/gettytab** file, a set of system-supplied defaults is used.

When the indicated entry is located, **getty** clears the terminal screen, prints a banner heading, and prompts for a login name. Usually, either the banner or the login prompt includes the system's hostname.

Next, **getty** prompts for a login and reads the login name, one character at a time. When it receives a null character (which is assumed to be the result pressing the BREAK , or "interrupt" key), **getty** switches to the entry **gettytab** entry named in the **nx** field. It reinitializes the line to the new characteristics, and then prompts for a login once again. This mechanism typically is used to cycle through a set of line speeds (baud rates) for each terminal line. For instance, a rotary dialup might have entries for the speeds: 300, 1200, 150, and 110 baud, with each **nx** field pointing to the next one in succession.

The user terminates login input line with a NEWLINE or RETURN character. The latter is preferable; it sets up the proper treatment of RETURN characters (see **tty**(4)). **getty** checks to see if the terminal has only upper-case alphabetical characters. If all alphabetical characters in the login name are in upper case, the system maps them along with all subsequent upper-case input characters to lower-case internally; they are displayed in upper case for the benefit of the terminal. To force recognition of an upper-case character, the shell allows them to be quoted (typically by preceding each with a backslash, '\').

Finally, **getty** calls **login**(1) with the login name as an argument.

getty can be set to time out after a certain interval; this hangs up dial-up lines if the login name is not entered in time.

Sun386i SYSTEM DESCRIPTION

For Sun386i system, the value of *type* is the constant **Sun**, for the console frame buffer.

Sun386i SYSTEM OPTIONS

–n invoke the full screen login program **logintool**(8), and optionally the "New User Accounts" feature. May only be used on a frame buffer. Unless removed from the console entry in */etc/ttytab*, this option is in effect by default.

FILES

/etc/gettytab

SEE ALSO

login(1), **ioctl**(2), **tty**(4), **fbtab**(5), **gettytab**(5), **svdtab**(5), **ttytab**(5), **init**(8), **logintool**(8)

DIAGNOSTICS

***ttyxx*: No such device or address.**

***ttyxx*: No such file or directory.**

A terminal which is turned on in the **ttys** file cannot be opened, likely because the requisite lines are either not configured into the system, the associated device was not attached during boot-time system configuration, or the special file in **/dev** does not exist.

NAME

gpconfig – initialize the Graphics Processor

SYOPNSIS

/usr/etc/gpconfig *gpunit* [[**–b**] [**–f**] *fbunit*... [**–u** *microcode-file*]]

DESCRIPTION

gpconfig binds **cgtwo** frame buffers to the GP, (Graphics Processor) and loads and starts the appropriate microcode in the GP. For example, the command line:

/usr/etc/gpconfig gpone0 cgtwo0 cgtwo1

will bind the frame buffer boards **cgtwo0** and **cgtwo1** to the Graphics Processor **gpone0**. The devices **/dev/gpone0a** and **/dev/gpone0b** will then refer to the combination of **gpone** and **cgtwo0** or **cgtwo1** respectively.

The same **cgtwo** frame buffer cannot be bound to more than one GP.

All **cgtwo** frame buffer boards bound to a GP must be configured to the same width and height.

The standard version of the file **/etc/rc.local** contains the following **gpconfig** command line:

/usr/etc/gpconfig gpone0 –f –b cgtwo0

This binds **gpone0** and **cgtwo0** as **gpone0a**, causes **gpone0a** to use the Graphics Buffer Board if it is present, and redirects **/dev/fb** to be **/dev/gpone0a**. If another configuration is desired, edit the command line in **/etc/rc.local** to do the appropriate thing.

It is inadvisable to run the **gpconfig** command while the GP is being used. Unpredictable results may occur. If it is necessary to change the frame buffer bindings to the GP (or to stop using the GP altogether), bring the system down gently, boot single user, edit the **gpconfig** line in the **/etc/rc.local** file, and bring the system back up multiuser.

OPTIONS

–b Configure the GP to use the Graphics Buffer as well. Currently only one GP-to-frame-buffer binding is allowed to use the graphics buffer at a time. Only the last **–b** option in the command line takes effect.

–f Redirect **/dev/fb** to the device formed by binding *gpunit* with **fbunit**. Only the last **–f** option in the command line takes effect.

–u *microcode-file*
Load the specified microcode file instead of the default file from **/usr/lib**.

FILES

/dev/cgtwo[0-9]
/dev/fb
/dev/gpone[0-3][abcd]
/usr/lib/gp1cg2.1024.ucode
/usr/lib/gp1cg2.1152.ucode
/etc/rc.local

SEE ALSO

cgtwo(4S), **gpone**(4S)

NAME

grpck – check group database entries

SYNOPSIS

/usr/etc/grpck [*filename*]

AVAILABILITY

This command is available with the *System V* software installation option. Refer to *Installing SunOS 4.1* for information on how to install optional software.

DESCRIPTION

grpck checks that a file in **group**(5) does not contain any errors; it checks the **/etc/group** file by default.

FILES

/etc/group

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 lower-case 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.

First char in group name not lower case alpha
The group name in an entry does not begin witha lower-case letter.

Group name too long
The group name in an entry has more than 8 characters.

SEE ALSO

groups(1), **group**(5), **passwd**(5)

NAME

gxtest – stand alone test for the Sun video graphics board

SYNOPSIS

b /stand/gxtest

DESCRIPTION

gxtest runs stand alone, not under control of the operating system. With the PROM resident monitor in control of the system, type the command:

> b /stand/gxtest

and the monitor boots the video test program into memory. **gxtest** is completely self-explanatory and runs under its own steam. It reports any errors it finds on the screen.

NAME

halt – stop the processor

SYNOPSIS

/usr/etc/halt [**–nqy**]

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**(8), 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

–n Prevent the *sync* before stopping.

–q Do a quick halt. No graceful shutdown is attempted.

–y Halt the system, even from a dialup terminal.

FILES

/var/adm/wtmp login accounting file

SEE ALSO

reboot(8), **shutdown**(8), **syslogd**(8)

NAME

hostrfs – convert IP addresses to RFS format

SYNOPSIS

hostrfs *hostname* [*portnum*]

AVAILABILITY

This program is available with the *RFS* software installation option. Refer to *Installing SunOS 4.1* for information on how to install optional software.

DESCRIPTION

hostrfs converts IP addresses to a format suitable for use by Remote File Sharing (RFS). It takes a hostname and an optional portnumber and produces an address in the following format:

\x<AF-INET><*portnum*><*IP-address*>0000000000000000

Each field given above is a hex ASCII representation. The AF_INET field is the address family which always has the value **0002**. *portnum* is the two-byte TCP port number; if not specified on the command line it defaults to **1450**. *IP-address* is the IP address of the *hostname* given on the command line followed by 16 trailing zeroes.

The output of this command may be directly used as the network address field for the address of an RFS name server in the **rfmaster**(5) file. It may also be used as input to the **nlsadmin** (8) command to initialize the addresses on which the **listener** program listens for service requests.

EXAMPLES

The output of

example% hostrfs wopr

is

\000214508190350900000000000000000

The output of the command can be used to initialize the network address on which the RFS **listener** program listens for remote service requests, for example:

example# nlsadmin –l 'hostrfs wopr' tcp

SEE ALSO

rfmaster(5), **nlsadmin**(8)

System and Network Administration

NAME

htable – convert DoD Internet format host table

SYNOPSIS

/usr/etc/htable *filename*

DESCRIPTION

htable converts a host table in the format specified by RFC 952 to the format used by the network library routines. Three files are created as a result of running **htable**: **hosts**, **networks**, and **gateways**. The **hosts** file is used by the **gethostent**(3N) routines in mapping host names to addresses. The **networks** file is used by the **getnetent**(3N) routines in mapping network names to numbers. The **gateways** file is used by the routing daemon in identifying "passive" Internet gateways; see **routed**(8C) for an explanation.

If any of the files **localhosts**, **localnetworks**, or **localgateways** are present in the current directory, the file's contents is prepended to the output file without interpretation. This allows sites to maintain local aliases and entries which are not normally present in the master database.

htable is best used in conjunction with the **gettable**(8C) program which retrieves the DoD Internet host table from a host.

FILES

localhosts
localnetworks
localgateways

SEE ALSO

intro(3), **gethostent**(3N), **getnetent**(3N), **gettable**(8C), **routed**(8C)

Harrenstien, Ken, Mary Stahl, and Elizabeth Feinler, *DoD Internet Host Table Specification*, RFC 952, Network Information Center, SRI International, Menlo Park, Calif., October 1985.

BUGS

Does not properly calculate the **gateways** file.

NAME

icheck – file system storage consistency check

SYNOPSIS

/usr/etc/icheck [**–s**] [**–b** *numbers*] [*filesystem*]

DESCRIPTION

Note: **icheck** has been superseded for normal consistency checking by **fsck**(8).

icheck examines a file system, builds a bit map of used blocks, and compares this bit map against the free list maintained on the file system. The normal output of **icheck** includes a report of

The total number of files and the numbers of regular, directory, block special and character special files.

The total number of blocks in use and the numbers of single-, double-, and triple-indirect blocks and directory blocks.

The number of free blocks.

The number of blocks missing; that is, not in any file nor in the free list.

With the **–s** option **icheck** ignores the actual free list and reconstructs a new one by rewriting the super-block of the file system. The file system should be dismounted while this is done; if this is not possible (for example if the root file system has to be salvaged) care should be taken that the system is quiescent and that it is rebooted immediately afterwards so that the old, bad in-core copy of the superblock will not continue to be used. Notice also that the words in the superblock which indicate the size of the free list and of the i-list are believed. If the superblock has been curdled these words will have to be patched. The **–s** option suppresses the normal output reports.

Following the **–b** option is a list of block numbers; whenever any of the named blocks turns up in a file, a diagnostic is produced.

icheck is faster if the raw version of the special file is used, since it reads the i-list many blocks at a time.

SEE ALSO

fs(5), **clri**(8), **dcheck**(8), **fsck**(8), **ncheck**(8)

DIAGNOSTICS

For duplicate blocks and bad blocks (which lie outside the file system) **icheck** announces the difficulty, the i-number, and the kind of block involved. If a read error is encountered, the block number of the bad block is printed and **icheck** considers it to contain 0.

Bad freeblock
means that a block number outside the available space was encountered in the free list.

***n* dups in free**
means that *n* blocks were found in the free list which duplicate blocks either in some file or in the earlier part of the free list.

BUGS

Since **icheck** is inherently two-pass in nature, extraneous diagnostics may be produced if applied to active file systems.

It believes even preposterous superblocks and consequently can get core images.

The system should be fixed so that the reboot after fixing the root file system is not necessary.

NAME

idload – RFS user and group mapping

SYNOPSIS

idload [**–n**] [**–g** *g_rules*] [**–u** *u_rules*] [*directory*]

AVAILABILITY

This program is available with the *RFS* software installation option. Refer to *Installing SunOS 4.1* for information on how to install optional software.

DESCRIPTION

idload is used on Remote File Sharing (RFS) servers to build translation tables for user and group IDs. It takes your **/etc/passwd** and **/etc/group** files and produces translation tables for user and group IDs from remote machines, according to the rules set down in the *u_rules* and *g_rules* files. If you are mapping by user and group name, you will need copies of remote **/etc/passwd** and **/etc/group** files. If no rules files are specified, remote user and group IDs are mapped to MAXUID+1. This is an ID number that is one higher than the highest number you could assign on your system.

By default, the remote password and group files are assumed to reside in **/usr/nserve/auth.info**/*domain*/*host*/**[passwd | group]**. The *directory* argument indicates that some directory structure other than **/usr/nserve/auth.info** contains the *domain*/*host* **passwd** and **group** files. *host* is the name of the host the files are from and *domain* is the domain where *host* can be found.

This command is restricted to the super-user.

This command is run automatically when the first remote mount is done of a remote resource (see **mount**(8)).

If any of the following are true, an error message will be sent to standard error.

- Neither rules files can be found or opened.
- There are syntax errors in the rules file.
- There are semantic errors in the rules file.
- Host information could not be found.
- The command is not run with super-user privileges.

Partial failures will display a warning message, although the process will continue.

OPTIONS

–n Do not produce a translation table, however, send a display of the ID mapping to the standard out. This is used to do a trial run of the mapping.

–u *u_rules* The *u_rules* file contains the rules for user ID translation. The default rules file is **/usr/nserve/auth.info/uid.rules**.

–g *g_rules* The *g_rules* file contains the rules for group ID translation. The default rules file is **/usr/nserve/auth.info/gid.rules**.

USAGE

Rules

The rules files have two types of sections, both optional: **global** and **host**. There can be only one global section, though there can be one host section for each host you want to map.

The **global** section describes the default conditions for translation for any machines that are not explicitly referenced in a **host** section. If the global section is missing, the default action is to map all remote user and group IDs from undefined hosts to MAXUID+1. The syntax of the first line of the **global** section is:

global

A **host** section is used for each client machine or group of machines that you want to map differently from the global definitions. The syntax of the first line of each **host** section is:

host*name*[...]

where *name* is replaced by the full name(s) of a host (*domain.hostname*).

The format of a rules file is described below. All lines are optional, but must appear in the order shown.

global
default *local* | **transparent**
exclude
[*remote_id–remote_id*] | [*remote_id*]
map [*remote_id*:*local*]

host *domain.hostname* [*domain.hostname*...]
default *local* | **transparent**
exclude [*remote_id–remote_id*] | [*remote_id*] | [*remote_name*]
map [*remote*:*local*] | *remote* | **all**

Each of these instruction types is described below.

The line

default *local* | **transparent**

defines the mode of mapping for remote users that are not specifically mapped in instructions in other lines. **transparent** means that all remote user and group IDs will have the same numeric value locally unless they appear in the **exclude** instruction. *local* can be replaced by a local user name or ID to map all users into a particular local name or ID number. If the default line is omitted, all users that are not specifically mapped are mapped into a "special guest" login ID .

The line

exclude [*remote_id–remote_id*] | [*remote_id*] | [*remote_name*]

defines remote IDs that will be excluded from the **default** mapping. The **exclude** instruction must precede any **map** instructions in a block. You can use a range of ID numbers, a single ID number, or a single name. (*remote_name* cannot be used in a global block.)

The line

map [*remote*:*local*] | *remote* | **all**

defines the local IDs and names that remote IDs and names will be mapped into. *remote* is either a remote ID number or remote name; *local* is either a local ID number or local name. Placing a colon between a *remote* and a *local* will give the value on the left the permissions of the value on the right. A single *remote* name or ID will assign the user or group permissions of the same local name or ID. **all** is a predefined alias for the set of all user and group IDs found in the local **/etc/passwd** and **/etc/group** files. You cannot map by remote name in **global** blocks.

Note: **idload** will always output warning messages for '**map all**', since password files always contain multiple administrative user names with the same ID number. The first mapping attempt on the ID number will succeed, all subsequent attempts will fail.

RFS does not need to be running to use **idload**.

EXIT STATUS

On successful completion, **idload** will produce one or more translation tables and return a successful exit status. If **idload** fails, the command will return an unsuccessful exit status without producing a translation table.

FILES

/etc/passwd
/etc/group
/usr/nserve/auth.info/*domain***/***host***/[user | group]**
/usr/nserve/auth.info/vid.rules
/usr/nserve/auth.info/gid.rules

SEE ALSO

mount(8)

NAME

ifconfig – configure network interface parameters

SYNOPSIS

/usr/etc/ifconfig *interface* [*address_family*] [*address* [*dest_address*]] [**netmask** *mask*]
[**broadcast** *address*] [**up**] [**down**] [**trailers**] [**–trailers**] [**arp**] [**–arp**] [**private**]
[**–private**] [**metric** *n*]

/usr/etc/ifconfig *interface* [*protocol_family*]

DESCRIPTION

ifconfig is used to assign an address to a network interface and/or to configure network interface parameters. **ifconfig** must be used at boot time to define the network address of each interface present on a machine; it may also be used at a later time to redefine an interface's address or other operating parameters. Used without options, **ifconfig** displays the current configuration for a network interface. If a protocol family is specified, **ifconfig** will report only the details specific to that protocol family. Only the super-user may modify the configuration of a network interface.

The *interface* parameter is a string of the form *nameunit*, for example **ie0**. The interface name "*–a*" is reserved, and causes the remainder of the arguments to be applied to each address of each interface in turn.

Since an interface may receive transmissions in differing protocols, each of which may require separate naming schemes, the parameters and addresses are interpreted according to the rules of some address family, specified by the *address_family* parameter. The address families currently supported are **ether** and **inet**. If no address family is specified, **inet** is assumed.

For the TCP/IP family (**inet**), the address is either a host name present in the host name data base (see **hosts**(5)) or in the Network Interface Service (NIS) map **hosts**, or a TCP/IP address expressed in the Internet standard "dot notation". Typically, an Internet address specified in dot notation will consist of your system's network number and the machine's unique host number. A typical Internet address is **192.9.200.44**, where **192.9.200** is the network number and **44** is the machine's host number.

For the **ether** address family, the address is an Ethernet address represented as *x*:*x*:*x*:*x*:*x*:*x* where *x* is a hexadecimal number between 0 and ff. Only the super-user may use the **ether** address family.

If the *dest_address* parameter is supplied in addition to the *address* parameter, it specifies the address of the correspondent on the other end of a point to point link.

OPTIONS

up Mark an interface "up". This happens automatically when setting the first address on an interface. The **up** option enables an interface after an **ifconfig down**, reinitializing the hardware.

down Mark an interface "down". When an interface is marked "down", the system will not attempt to transmit messages through that interface. If possible, the interface will be reset to disable reception as well. This action does not automatically disable routes using the interface.

trailers This flag used to cause a non-standard encapsulation of inet packets on certain link levels. Sun drivers no longer use this flag, but it is ignored for compatibility.

–trailers Disable the use of a "trailer" link level encapsulation.

arp Enable the use of the Address Resolution Protocol in mapping between network level addresses and link level addresses (default). This is currently implemented for mapping between TCP/IP addresses and 10Mb/s Ethernet addresses.

–arp Disable the use of the Address Resolution Protocol.

private Tells the **in.routed** routing daemon (see **routed**(8C)) that the interface should not be advertised.

–private Specify unadvertised interfaces.

metric *n* Set the routing metric of the interface to *n*, default 0. The routing metric is used by the routing protocol (**routed**(8C)). Higher metrics have the effect of making a route less favorable; metrics are counted as addition hops to the destination network or host.

netmask mask (**inet** only) Specify how much of the address to reserve for subdividing networks into sub-networks. The mask includes the network part of the local address and the subnet part, which is taken from the host field of the address. The mask can be specified as a single hexadecimal number with a leading 0x, with a dot-notation address, or with a pseudo-network name listed in the network table **networks**(5). The mask contains 1's for the bit positions in the 32-bit address which are to be used for the network and subnet parts, and 0's for the host part. The mask should contain at least the standard network portion, and the subnet field should be contiguous with the network portion. If a '+' (plus sign) is given for the netmask value, then the network number is looked up in the NIS **netmasks.byaddr** map (or in the **/etc/netmasks**) file if not running the NIS service.

broadcast *address*
(**inet** only) Specify the address to use to represent broadcasts to the network. The default broadcast address is the address with a host part of all 0's. A + (plus sign) given for the broadcast value causes the broadcast address to be reset to a default appropriate for the (possibly new) address and netmask. Note that the arguments of **ifconfig** are interpreted left to right, and therefore

ifconfig –a netmask + broadcast +

and

ifconfig –a broadcast + netmask +

may result in different values being assigned for the interfaces' broadcast addresses.

EXAMPLES

If your workstation is not attached to an Ethernet, the **ie0** interface should be marked "down" as follows:

ifconfig ie0 down

To print out the addressing information for each interface, use

ifconfig –a

To reset each interface's broadcast address after the netmasks have been correctly set, use

ifconfig –a broadcast +

FILES

/dev/nit
/etc/netmasks

SEE ALSO

intro(3), **ethers**(3N), **arp**(4P), **hosts**(5), **netmasks**(5), **networks**(5) **netstat**(8C), **rc**(8), **routed**(8C).

DIAGNOSTICS

Messages indicating the specified interface does not exist, the requested address is unknown, or the user is not privileged and tried to alter an interface's configuration.

NOTES

The Network Information Service (NIS) was formerly known as Sun Yellow Pages (YP). The functionality of the two remains the same; only the name has changed.

NAME

imemtest – stand alone memory test

SYNOPSIS

b /stand/imemtest

DESCRIPTION

imemtest runs stand alone, not under control of the operating system. With the PROM resident monitor in control of the system, type the command:

> b /stand/imemtest

and the monitor boots the memory test program into memory. **imemtest** is completely self-explanatory. It prompts for all start and end addresses, and after that it runs under its own steam. It reports any errors it finds on the screen.

NAME

inetd – Internet services daemon

SYNOPSIS

/usr/etc/inetd [**–d**] [*configuration-file*]

DESCRIPTION

inetd, the Internet services daemon, is normally run at boot time by the **/etc/rc.local** script. When started **inetd** reads its configuration information from *configuration-file*, the default being **/etc/inetd.conf**. See **inetd.conf**(5) for more information on the format of this file. It listens for connections on the Internet addresses of the services that its configuration file specifies. When a connection is found, it invokes the server daemon specified by that configuration file for the service requested. Once a server is finished, **inetd** continues to listen on the socket (except in some cases which will be described below).

Depending on the value of the "wait-status" field in the configuration line for the service, **inetd** will either wait for the server to complete before continuing to listen on the socket, or immediately continue to listen on the socket. If the server is a "single-threaded" datagram server (a "wait-status" field of "wait"), **inetd** must wait. That server will handle all datagrams on the socket. All other servers (stream and ×lti-threaded" data-gram, a "wait-status" field of "nowait") operate on separate sockets from the connection request socket, thus freeing the listening socket for new connection requests.

Rather than having several daemon processes with sparsely distributed requests each running concurrently, **inetd** reduces the load on the system by invoking Internet servers only as they are needed.

inetd itself provides a number of simple TCP-based services. These include **echo**, **discard**, **chargen** (character generator), **daytime** (human readable time), and **time** (machine readable time, in the form of the number of seconds since midnight, January 1, 1900). For details of these services, consult the appropriate RFC, as listed below, from the Network Information Center.

inetd rereads its configuration file whenever it receives a hangup signal, **SIGHUP**. New services can be activated, and existing services deleted or modified in between whenever the file is reread.

SEE ALSO

inetd.conf(5), **comsat**(8C), **ftpd**(8C), **rexecd**(8C), **rlogind**(8C), **rshd**(8C), **telnetd**(8C), **tftpd**(8C)

Postel, Jon, *Echo Protocol*, RFC 862, Network Information Center, SRI International, Menlo Park, Calif., May 1983.

Postel, Jon, *Discard Protocol*, RFC 863, Network Information Center, SRI International, Menlo Park, Calif., May 1983.

Postel, Jon, *Character Generator Protocol*, RFC 864, Network Information Center, SRI International, Menlo Park, Calif., May 1983.

Postel, Jon, *Daytime Protocol*, RFC 867, Network Information Center, SRI International, Menlo Park, Calif., May 1983.

Postel, Jon, and Ken Harrenstien, *Time Protocol*, RFC 868, Network Information Center, SRI International, Menlo Park, Calif., May 1983.

NAME

infocmp – compare or print out terminfo descriptions

SYNOPSIS

infocmp [**–cdnILCruvV1**] [**–sd**] [**–si**] [**–sl**] [**–sc**] [**–w** *width*] [**–A** *directory*] [**–B** *directory*] [*termname* ...]

SYNOPSIS

/usr/5bin/infocmp *arguments*

Note: *arguments* to **/usr/5bin/infocmp** are the same as those for **infocmp**, above.

AVAILABILITY

The System V version of this command is available with the *System V* software installation option. Refer to *Installing SunOS 4.1* for information on how to install optional software.

DESCRIPTION

infocmp compares a binary **terminfo**(5V) entry with other terminfo entries, rewrites a **terminfo** description to take advantage of the **use=***field*, or prints out a **terminfo** description from the corresponding binary file in a variety of formats. It displays boolean fields first, then numeric fields, then string fields.

It can also convert a **terminfo** entry to a **termcap**(5) entry; the **–C** flag causes **infocmp** to perform this conversion. Some **termcap** variables are not supported by **terminfo**, but those that can be derived from *terminfo* variables are displayed. Not all **terminfo** capabilities are translated either; only those that are allowed in a **termcap** entry are normally displayed. Specifying the **–r** option eliminates this restriction, allowing all capabilities to be displayed in **termcap** form.

Because padding is collected at the beginning of a capability, not all capabilities are displayed. Since mandatory padding is not supported by **terminfo** and **termcap** strings are not as flexible, it is not always possible to convert a **terminfo** string capability into an equivalent working **termcap** capability. Also, a subsequent conversion of the **termcap** file back into **terminfo** format will not necessarily reproduce the original source; **infocmp** attempts to convert parameterized strings, and comments out those that it can not.

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

If no *termname* arguments are given, the environment variable TERM is used for all expected *termname* arguments.

OPTIONS

Default Options

If no options are specified and either zero or one *termname* is specified, the **–I** option is assumed to be in effect. If more than one *termname* is specified, the **–d** option is assumed.

Comparison Options

infocmp compares the description of the first terminal *termname* with each of the descriptions for terminals listed in subsequent *termname* arguments. 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.

–c Produce a list of capabilities common to both 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.

-d Produce a list of capabilities that differ between descriptions.

-n Produce a list of capabilities in neither entry.

Source Listing Options

The **-I**, **-L**, and **-C** options 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 Display only those capabilities that have **termcap** equivalents, using the **termcap** names and displaying them in **termcap** form whenever possible.

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. All padding information for strings is collected together and placed at the beginning of the string where **termcap** expects it. Mandatory padding (padding information with a trailing '/') will become optional.

-r When using **-C**, display all capabilities, not just those capabilities that have **termcap** equivalents.

-u Produce a **terminfo** source description for the first named terminal which is relative to the descriptions given by the entries for all terminals named subsequently on the command line, by analyzing the differences between them, 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 is displayed with an at-sign (@) if it no longer exists in the first terminal, but one of the other terminal 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 has a different value for that capability.

The order of the other *termname* entries is significant. Since the terminfo compiler **tic**(8V) 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 in which they are given. **infocmp** flags any such inconsistencies between the other *termname* entries as they are found.

Alternatively, specifying a capability after a **use=** entry that contains it, 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.

Specifying superfluous **use=** slows down the comparison, but is not fatal; **infocmp** flags superfluous **use=** fields.

Sorting Options

-sd Sort fields in the order that they are stored in the **terminfo** database.

-si Sort fields by **terminfo** name.

-sl Sort fields by the long C variable name.

-sc Sort fields by the **termcap** name.

If no sorting option is given, fields are 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.

Changing Databases

The location of the compiled **terminfo** database is taken from the environment variable TERMINFO. If the variable is not defined, or if the terminal is not found in that location, the system **terminfo** database, usually in **/usr/share/lib/terminfo**, is used. The options **–A** and **–B** may be used to override this location. With these options, 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.

–A Set TERMINFO for the first *termname* argument.

–B Set TERMINFO for the remaining *termname* arguments.

Other Options

–v Print out tracing information on the standard error.

–V Print out the version of the program in use on the standard error and exit.

–1 Print fields out one to a line. Otherwise, fields are printed several to a line to a maximum width of 60 characters.

–w *width*
Change the output to *width* characters.

FILES

/usr/share/lib/terminfo/?/*
compiled terminal description database

/usr/5include/term.h

SEE ALSO

curses(3V), **termcap**(5), **terminfo**(5V), **tic**(8V)

DIAGNOSTICS

malloc is out of space!
There was not enough memory available to process all the terminal descriptions requested. Run **infocmp** in several smaller stages (with fewer *termname* arguments).

use= order dependency found:
A value specified in one relative terminal specification was different from that in another relative terminal specification.

'use=*term*' did not add anything to the description.
A relative terminal name did not contribute anything to the final description.

must have at least two terminal names for a comparision to be done.
The **–u**, **–d** and **–c** options require at least two terminal names.

NAME

init – process control initialization

SYNOPSIS

/usr/etc/init [**–bs**]

DESCRIPTION

init is invoked inside the operating system as the last step in the boot procedure. It normally runs the sequence of commands in the script **/etc/rc.boot** (see **rc**(8)) to check the file system. If passed the **–b** option from the boot program, **init** skips this step. If the file system check succeeds or is skipped, **init** runs the commands in **/etc/rc** and **/etc/rc.local** to begin multiuser operation; otherwise it commences single-user operation by giving the super-user a shell on the console. It is possible to pass the –s parameter from the boot program to **init** so that single-user operation is commenced immediately.

Whenever a single-user shell is created, and the system is running as a secure system, the **init** program demands the super-user password. This is to prevent an ordinary user from invoking a single-user shell and thereby circumventing the system's security. Logging out (for instance, by entering an EOT) causes **init** to proceed with a multi-user boot. The super-user password is demanded whenever the system is running secure as determined by **issecure**(3), or the console terminal is not labeled "secure" in **/etc/ttytab**.

Whenever single-user operation is terminated (for instance by killing the single-user shell) **init** runs the scripts mentioned above.

In multi-user operation, **init**'s role is to create a process for each terminal port on which a user may log in. To begin such operations, it reads the file **/etc/ttytab** and executes a command for each terminal specified in the file. This command will usually be **/usr/etc/getty**. **getty**(8) opens and initializes the terminal line, reads the user's name and invokes **login**(1) to log in the user and execute the shell.

Ultimately the shell will terminate because it received EOF, either explicitly, as a result of hanging up, or from the user logging out. The main path of **init**, which has been waiting for such an event, wakes up and removes the appropriate entry from the file **/etc/utmp**, which records current users. **init** then makes an entry in **/var/adm/wtmp**, which maintains a history of logins and logouts. The **/var/adm/wtmp** entry is made only if a user logged in successfully on the line. Then the appropriate terminal is reopened and the command for that terminal is reinvoked.

init catches the *hangup* signal (SIGHUP) and interprets it to mean that the file **/etc/ttytab** should be read again. The shell process on each line which used to be active in **/etc/ttytab** but is no longer there is terminated; a new process is created for each added line; lines unchanged in the file are undisturbed. Thus it is possible to drop or add terminal lines without rebooting the system by changing **/etc/ttytab** and sending a *hangup* signal to the **init** process: use '**kill –HUP 1**'.

init terminates multi-user operations and resumes single-user mode if sent a terminate (SIGTERM) signal: use '**kill –TERM 1**'. If there are processes outstanding which are deadlocked (due to hardware or software failure), **init** does not wait for them all to die (which might take forever), but times out after 30 seconds and prints a warning message.

init ceases to create new processes, and allows the system to slowly die away, when sent a terminal stop (SIGTSTP) signal: use '**kill –TSTP 1**'. A later hangup will resume full multi-user operations, or a terminate will initiate a single-user shell. This hook is used by **reboot**(8) and **halt**(8).

Whenever it reads **/etc/ttytab**, **init** will normally write out an old-style **/etc/ttys** file reflecting the contents of **/etc/ttytab**. This is required in order that programs built on earlier versions of SunOS that read the **/etc/ttys** file (for example, programs using the **ttyslot**(3V) routine, such as **shelltool** (1)) may continue to run. If it is not required that such programs run, **/etc/ttys** may be made a link (hard or symbolic) to **/etc/ttytab** and **init** will not write to **/etc/ttys**.

init's role is so critical that if it dies, the system will reboot itself automatically. If, at bootstrap time, the **init** program cannot be located, the system will print an error message and panic.

FILES

/dev/console
/dev/tty*
/etc/utmp
/var/adm/wtmp
/etc/ttytab
/etc/rc
/etc/rc.local
/etc/rc.boot
/usr/etc/getty

SEE ALSO

kill(1), **login**(1), **sh**(1), **shelltool**(1), **issecure**(3), **ttyslot**(3V), **ttytab**(5), **getty**(8), **halt**(8), **rc**(8), **reboot**(8), **shutdown**(8)

DIAGNOSTICS

command **failing, sleeping.**

A process being started to service a line is exiting quickly each time it is started. This is often caused by a ringing or noisy terminal line. **init** will sleep for 30 seconds, then continue trying to start the process.

WARNING: Something is hung (won't die); ps axl advised.

A process is hung and could not be killed when the system was shutting down. This is usually caused by a process which is stuck in a device driver due to a persistent device error condition.

NAME

installboot – install bootblocks in a disk partition

SYNOPSIS

/usr/mdec/installboot [**–lvt**] *bootfile protobootblk bootdevice*

DESCRIPTION

The **boot**(8S) program is loaded from disk by bootblock code which resides in the bootblock area of a disk partition. In order for the bootblock code to read the boot program (usually **/boot**) it is necessary for it to know the block numbers occupied by the boot program. Previous versions of the bootblock code could find **/boot** by interpreting the file system on the partition from which it was being booted, but this is no longer so.

installboot plugs the block numbers of the boot program into a table in the bootblock code, and writes the modified bootblock code onto the disk. Note: **installboot** must be run every time the boot program is reinstalled, since in general, the block list of the boot program will change each time it is written.

bootfile is the name of the boot program, usually **/boot**. *protobootblk* is the name of the bootblock code into which the block numbers of the boot program are to be inserted. The file read in must have an **a.out**(5) header, but it will be written out to the device with the header removed. *bootdevice* is the name of the disk device onto which the bootblock code is to be installed.

OPTIONS

–l Print out the list of block numbers of the boot program.

–t Test. Display various internal test messages.

–v Verbose. Display detailed information about the size of the boot program, etc.

EXAMPLE

To install the bootblocks onto the root partition on a Xylogics disk:

```
example% cd /usr/mdec
example% installboot –vlt /boot bootxy /dev/rxy0a
```

For an SD disk, you would use **bootsd** and **/dev/rsd0a**, respectively, in place of **bootxy** and **/dev/rxy0a**.

SEE ALSO

od(1V), **a.out**(5) **boot**(8S), **bootparamd**(8), **init**(8), **kadb**(8S), **monitor**(8S), **ndbootd**(8C), **rc**(8), **reboot**(8)

System and Network Administration

Installing SunOS 4.1

NAME

install_small_kernel – install a small, pre-configured kernel

SYNOPSIS

/usr/etc/install/install_small_kernel [*hostname*] ...

DESCRIPTION

install_small_kernel is a script that installs a small, pre-configured kernel, **GENERIC_SMALL** on a host. This kernel supports approximately four users, and is only available for the following configurations:

Sun-3/50 and Sun-3/60 systems with up to 2 SCSI disks, 1 SCSI tape
Sun-3/80 systems with up to 4 SCSI disks, 1 SCSI tape
Sun-4/110 systems with up to 2 SCSI disks, 1 SCSI tape
SPARCsystem 330 systems with up to 4 SCSI disks, 1 SCSI tape
SPARCstation 1 systems with up to 4 SCSI disks. 1 floppy drive and 2 SCSI tapes

If *hostname* is a server that does not fit any of the above configurations, **install_small_kernel** can be used to install the small kernel on its clients.

If no hostnames are specified, **install_small_kernel** cycles through all the clients configured for a server to determine the small kernel installs to be made. If the '**small_kernel**' flag in the client file, **/etc/install/client.***hostname* is set to '**yes**', that client will not be processed. To force re-installation of a small kernel on any clients, simply call **install_small_kernel** with the appropriate client names.

install_small_kernel prompts for confirmation before actually doing the install on any host.

install_small_kernel is executable from the miniroot, as well as single-user and multi-user modes. It supports standalone and server configuration in all cases, but dataless systems are supported in multi-user mode only. This script is restricted to the super-user.

FILES

/usr/sys/sun*arch***/conf/GENERIC_SMALL**
kernel configuration file for *arch* **/usr/install/client.***hostname*

SEE ALSO

add_client(8), **add_services**(8), **rm_client**(8), **suninstall**(8)

System and Network Administration

NAME

installtxt, gencat – create a message archive

SYNOPSIS

/usr/etc/installtxt [[–]**d**|**c**|**r**|**t**|**x**|**i** [**ouvs**]]] *message-archive*... [*source-message-file*]

/usr/etc/gencat *catfile msgfile*...

DESCRIPTION

installtxt converts each *source-message-file* into a binary format message archive. At the same time, if necessary, **installtxt** maintains groups of files (member files) combined into a single message archive. **installtxt** is normally used to create and update message archives used by the run-time message handling facility **gettext**(3).

gencat performs the same function as **installtxt**, but supports the X/Open catalog source format.

installtxt creates the message archive in *message-archive*. If the message archive does not exist, it is created by the **–c** option. *source-message-file* contains source versions of the target strings. On successful completion of an update operation of **installtxt**, the message archive will have been updated with details of the formatted version of each *source-message-file*. If *message-archive* does not contain the full pathname of the run-time location of the message catalog, it will have to be moved to the appropriate locale directory before applications using the archive are activated.

gencat merges the message text source files (*msgfile*...) into a formatted message catalog *catfile*. *catfile* is created if it does not already exist. If *catfile* does exist, its messages are included in the new *catfile*. If set and message numbers collide, the new message-text defined in *msgfile* will replace the old message text currently contained in *catfile*. The output formats of both *message_archive* and *catfile* are the same. However it should be noted that on a per-application basis, it is not intended that the output forms of these two utilities should be mixed, and the consequence of doing so is undefined.

OPTIONS

The following options and modifiers apply to **installtxt** only. For **installtxt** you must indicate only one of: **c**, **d**, **r**, **t**, or, **x**, which may be followed by one or more **Modifiers**, **o**, **u**, or , **v**.

The options are:

c Create. The member file called *source-message-file* is being made for the first time in the message archive. It should not exist already.

d Delete the named member files from *message archive*. Note that individual messages can be deleted by entering an empty value after the message-id selecting the message to be deleted. With the **v** option these deletions are notified on the standard output.

r Replace the named member files in the message archive. This allows the existing *message archive* to be merged with new versions of messages. No new message will be added to the message archive unless each message-tag in the *source-message-file* is unique in the active domain. If the member file contains a message-tag that is not unique within the active domain, **installtxt** will fail and the contents of the active message archive will not be altered.

t Table of contents. Produces a list on the standard output of all member files in *message_archive*.

x Extract. If no names are given, all member files in the message archive are extracted into the current directory; if names are given, only those files are extracted. In neither case does **x** alter the message archive. The extracted member files will be returned in their original source format. It is possible for the **–x** option to lose comments that were contained in the original source message file. In addition, overlong lines may be escaped (using \n) at a point that is different from the original source, although the end result will logically be the same string.

Modifiers

o Old date. When member files are extracted with the **x** option, set the "last modified" date to the date recorded in the message archive.

u Update. Replace only those member files that have changed since they were put in the message archive. Used with the **r** option.

v Verbose. When used with the **c**, **r**, or **d** option, give a file-by-file description of the creation of a new *message archive* file from the old version and the constituent member files. When used with **x**, give a file-by-file description of the extraction of message archive member files. When used with **t**, print information about the size and creation date of the message archive, as well as a count of the number of target strings in the message-archive.

USAGE

source-message-file consists of one or more lines of text, with each line containing either a comment, a directive or a text line. The format of a comment line is:

"$ %s", *comment*

A line beginning with a dollar sign ($), followed by a *blank* character streated as a comment line. The format of directives is:

"$%s %s", *control-type*, *value*

Directives should be directly preceded by a dollar sign ($), and followed by an optional value. There is one *blank* character between the directive and its value. The following directives are recognized:

$separator *c*

This directive specifies an optional separator character that will subsequently be used in the following text lines to separate the message identifier from the target string. There is one *blank* character between **separator** and the separator character itself. If this line is absent then the default separator is the *blank* character. Only the first occurrence of this character on one text line will be interpreted, for example:

$separator :
12345:Bonjour: Mon ami

would declare the message identifier to be 12345, the target string would contain the second ":".

$domain *domain*

This directive states that all following target strings are contained within a domain of the object message file as described by *domain*. *domain* can be any string of up to {PATH_MAX} bytes in length.

$quote *c* This directive specifies an optional quote character *c*, which can be used to surround both *message_string* and *message_identifier* . By default, or if an empty **$quote** directive is supplied, no quoting of *message_string* will be recognized. If the **$quote** directive is given then all message strings must contain pairs of quotes, although quotes around the message_identifier are still optional after the directive.

The format of the text line is:

"%s%s%s", *message_identifier*, *separator_character*, *message_string*

Each line defines a message identifier and a target string pair.

Empty lines in a source text file are ignored. If a *message_identifier* starts with a dollar ($) character, then that dollar character must be escaped with a backslash (\$). Any other form of input line syntax is illegal and will cause **installtxt** to exit with the error value.

Message strings and message identifiers can contain the special characters and escape sequences as defined in the following table:

Description	Symbol
newline	**\n**
tab	**\t**
vertical-tab	**\v**
backspace	**\b**
carriage-return	**\r**
form-feed	**\f**
backslash	****
bit pattern	**\ddd**

The escape sequence *\ddd* consists of backslash followed by 1, 2 or 3 octal digits, which are used to specify the value of the desired character. If *message_identifier* contains the separator character then it must be escaped with a backslash (\) character. If the character following a backslash is not one of those specified, the effect is unspecified.

Backslash, \, followed by a NEWLINE character is used to continue an individual string on the following line. Both *message_identifier* and *message_string* may be continued over lines in this way. *message_string* is stored in *object_file* in an implementation specific way. If *message_string* is empty, and *separator* is present, a null string is stored in *object_file*.

msgfile must be in the X/Open **gencat** format.

EXAMPLES

```
# /bin/sh script
# The following creates a message archive in the file messages.general
installtxt –cv messages.general input
#
```

FILES

/etc/locale/LC_MESSAGES/*locale/domain*
: standard private location for message archive/catalog in locale *locale* and domain *domain*

/usr/share/lib/locale/LC_MESSAGES
: standard shared location for message archive/catalog in locale *locale* and domain *domain*

SEE ALSO

catgets(3), **gettext**(3), **setlocale**(3V), **locale**(5)

X/Open Portability Guide Issue 2

NAME

intr – allow a command to be interruptible

SYNOPSIS

intr [**–anv**] [**–t** *seconds*] *command* [*arguments*]

DESCRIPTION

intr executes *command* after altering the execution environment to make *command* to be interrutable.

Since interactive commands are by default interruptable, **intr** is intended for use as a wrapper around commands started by the **/etc/rc** files; commands spawned from these files are not interruptable by default. It has no other intended use than as a wrapper around **/etc/rc** commands.

The following signals are ignored as a result of wrapping **intr** around a command:

SIGTSTP	terminal generated stop signal
SIGTTIN	background read
SIGTTOU	background write

The following signals are reset to their default actions:

SIGINT	interrupt signal
SIGQUIT	quit signal

OPTIONS

–v	Echo the command in the form ' *command*' (note leading SPACE).
–a	Echo the command and its arguments.
–n	Do not echo a NEWLINE after the command or arguments (for example '**echo –n ...**').
–t *secs*	Arrange to have a **SIGALRM** signal delivered to the command in *secs* seconds.

EXAMPLES

All of these examples assume that they are in an **/etc/rc** file, that is, talking to the console, and not run interactively. The following example runs **fsck**(8) but allow it to be killed from the console:

intr fsck –p –w / /usr

Echoing is provided so that

ypbind; echo –n ' ypbind'

can be replaced with

intr –vn ypbind

Timeouts are provided so that the machine will not hang at boot:

intr –t 10 rdate *date_host*

SEE ALSO

echo(1V), **login**(1), **init**(8), **rc**(8)

BUGS

The –v option is a kludge.

NAME

iostat – report I/O statistics

SYNOPSIS

iostat [**–cdDIt**] [**–l** *n*] [*disk* ...] [*interval* [*count*]]

DESCRIPTION

iostat can iteratively report terminal and disk I/O activity, as well as CPU utilization. The first report is for all time since a reboot and each subsequent report is for the prior interval only.

In order to compute this information, the kernel maintains a number of counters. For each disk, seeks and data transfer completions and number of words transferred are counted; for terminals collectively, the number of input and output characters are counted. Also, at each clock tick, the state of each disk is examined and a tally is made if the disk is active. The kernel also provides approximate transfer rates of the devices.

OPTIONS

iostat's activity class options default to **tdc** (terminal, disk, and CPU). If any activity class options are specified, the default is completely overridden. Therefore, if only **–d** is specified, neither terminal nor CPU statistics will be reported. The last disk option specified (either **–d** or **–D**) is the only one that is used.

–c Report the percentage of time the system has spent in user mode, in user mode running low priority processes, see **nice**(1), in system mode, and idling.

–d For each disk, report the number of kilobytes transferred per second, the number of transfers per second, and the milliseconds per average seek (see BUGS below).

–D For each disk, report the reads per second, writes per second, and percentage disk utilization.

–I Report the counts in each interval, rather than reporting rates.

–t Report the number of characters read and written to terminals.

–l *n* Limit the number of disks included in the report to *n*; the disk limit defaults to 4. Note: disks explicitly requested (see *disk* below) are not subject to this disk limit.

disk Explicitly specify the disks to be reported; in addition to any explicit disks, any active disks up to the disk limit (see **–l** above) will also be reported.

interval Report once each *interval* seconds.

count Only print *count* reports.

FILES

/dev/kmem
/vmunix

SEE ALSO

vmstat(8)

BUGS

Milliseconds per average seek is an approximation based on the disk (not the controller) transfer rate. Therefore, the seek time will be over-estimated in systems with slower controllers.

NAME

ipallocd – Ethernet-to-IP address allocator

SYNOPSIS

/usr/etc/rpc.ipallocd

AVAILABILITY

Available only on Sun 386i systems running a SunOS 4.0.*x* release or earlier. Not a SunOS 4.1 release feature.

DESCRIPTION

ipallocd is a daemon that determines or temporarily allocates IP addresses within a network segment. The service is only available on the system which is home to the address authority for the network segment, currently the Network Interface Service (NIS) master of the **hosts.byaddr** map although the service is not tied to the NIS service. It has complete knowledge of the hosts listed in the NIS service, and, if the system is running the name server, of any hosts listed in internet domain tables automatically accessed on that host through the standard library **gethostent**(3N) call.

This protocol uses DES authentication (the Sun Secure RPC protocol) to restrict access to this function. The only clients privileged to allocate addresses are those whose net IDs are in the networks group. For machine IDs, the machine must be an NIS server.

The daemon uses permanent entries in the **/etc/ethers** and **/etc/hosts** files when they exist and are usable. In other cases, such as when a system is new to the network, **ipallocd** enters a temporary mapping in a local cache. Entries in the cache are removed when there have been no references to a given entry in the last hour. This cache survives system crashes so that IP addresses remain consistent.

The daemon also provides corresponding IP address to name mapping.

If the file **/etc/ipalloc.netrange** exists, **ipallocd** refuses to allocate addresses on networks not listed in the **netrange** file, or for which no free address is available.

FILES

/etc/ipalloc.cache	temporary cache
/etc/ipalloc.netrange	optional file to allocate network addresses

SEE ALSO

ipalloc(3R), **pnp**(3R), **ipalloc.netrange**(5), **ipallocd**(8C), **netconfig**(8C), **pnpboot**(8C), **rarpd**(8C)

NOTES

The Network Information Service (NIS) was formerly known as Sun Yellow Pages (YP). The functionality of the two remains the same; only the name has changed.

NAME

kadb – adb-like kernel and standalone-program debugger

SYNOPSIS

> b kadb [–d] [*boot-flags* **]**

DESCRIPTION

kadb is an interactive debugger that is similar in operation to **adb**(1), and runs as a standalone program under the PROM monitor. You can use **kadb** to debug the kernel, or to debug any standalone program.

Unlike **adb**, **kadb** runs in the same supervisor virtual address space as the program being debugged — although it maintains a separate context. The debugger runs as a *coprocess* that cannot be killed (no '**:k**') or rerun (no '**:r**'). There is no signal control (no '**:i**', '**:t**', or '**$i**'), although the keyboard facilities (CTRL–C, CTRL–S, and CTRL–Q) are simulated.

While the kernel is running under **kadb**, the abort sequence (**L1-A** or **BREAK**) drops the system into **kadb** for debugging — as will a system panic. When running other standalone programs under **kadb**, the abort sequence will pass control to the PROM monitor. **kadb** is then invoked from the monitor by jumping to the starting address for **kadb** found in **/usr/include/debug/debug.h** The following list gives the monitor commands to use for each system.

System	Monitor Command
Sun-2	**g fd00000**
Sun-3	**g fd00000**
Sun386i	**g fe005000**
Sun-4	**g ffc00000**
SPARCstation 1	**go ffc00000**

The **kadb** user interface is similar to that of **adb**. Note: **kadb** prompts with

kadb>

Most **adb** commands function in **kadb** as expected. Typing an abort sequence in response to the prompt returns you to the PROM monitor, from which you can examine control spaces that are not accessible within **adb** or **kadb**. The PROM monitor command **c** will return control to **kadb**. As with '**adb –k**', **$p** works when debugging kernels (by actually mapping in new user pages). The verbs **?** and **/** are equivalent in **kadb** , since there is only one address space in use.

OPTIONS

kadb is booted from the PROM monitor as a standalone program. If you omit the **–d** flag, **kadb** automatically loads and runs **vmunix** from the filesystem **kadb** was loaded from. The **kadb vmunix** variable can be patched to change the default program to be loaded.

–d Interactive startup. Prompts with

kadb:

for a file to be loaded. From here, you can enter a boot sequence line to load a standalone program. Boot flags entered in response to this prompt are included with those already set and passed to the program. If you type a RETURN only, **kadb** loads **vmunix** from the filesystem that **kadb** was loaded from.

boot-flags

You can specify boot flags as arguments when invoking **kadb.** Note: **kadb** always sets the **–d** (debug) boot flag, and passes it to the program being debugged.

USAGE

Refer to **adb** in *Debugging Tools.*

Kernel Macros

As with **adb**, kernel macros are supported. With **kadb**, however, the macros are compiled into the debugger itself, rather than being read in from the filesystem. The **kadb** command **$M** lists macros known to **kadb.**

Setting Breakpoints

Self-relocating programs such as the SunOS kernel need to be relocated before breakpoints can be used. To set the first breakpoint for such a program, start it with **':s'**; **kadb** is then entered after the program is relocated (when the system initializes its interrupt vectors). Thereafter, **':s'** single-steps as with **adb**. Otherwise, use **':c'** to start up the program.

Sun386i System Commands

The Sun386i system version of **kadb** has the following additional commands. Note, for the general syntax of **adb** commands, see **adb**(1).

:i	Read a byte (with the INB instruction) in from the port at *address*.
:o	Send a byte (with the OUTB instruction) containing *count* out through the port at *address*.
:p	Like **:b** in **adb**(1), but sets a breakpoint using the hardware debug register instead of the breakpoint instruction. The advantage of using **:p** is that when setting breakpoints with the debug register it is not necessary to have write access to the breakpoint location. Four (4) breakpoints can be set with the hardware debug registers.
$S	Switch I/O from the console to the serial port or vice versa.
[	Like **:e** in **adb**(1), but requires only one keystroke and no RETURN character.
]	Like **:s** in **adb**(1), but requires only one keystroke and no RETURN character.

Automatic Rebooting with kadb

You can set up your workstation to automatically reboot **kadb** by patching the *vmunix* variable in **/boot** with the string **kadb**. (Refer to **adb** in *Debugging Tools* for details on how to patch executables.)

FILES

/vmunix
/boot
/kadb
/usr/include/debug/debug.h

SEE ALSO

adb(1), **boot**(8S)

Debugging Tools
Writing Device Drivers

BUGS

There is no floating-point support, except on Sun386i systems.

kadb cannot reliably single-step over instructions that change the status register.

When sharing the keyboard with the operating system the monitor's input routines can leave the keyboard in a confused state. If this should happen, disconnect the keyboard momentarily and then reconnect it. This forces the keyboard to reset as well as initiating an abort sequence.

Most of the bugs listed in **adb**(1) also apply to **kadb**.

NAME

keyenvoy – talk to keyserver

SYNOPSIS

keyenvoy

DESCRIPTION

keyenvoy is used by some RPC programs to talk to the key server, **keyserv**(8C). The key server will not talk to anything but a root process, and **keyenvoy** is a set-uid root process that acts as an intermediary between a user process that wishes to talk to the key server and the key server itself.

This program cannot be run interactively.

SEE ALSO

keyserv(8C)

NAME

keyserv – server for storing public and private keys

SYNOPSIS

keyserv [**–dkn**]

DESCRIPTION

keyserv is a daemon that is used for storing the private encryption keys of each user logged into the system. These encryption keys are used for accessing secure network services such as secure NFS. When a user logs in to the system, the **login**(1) program uses the login password to decrypt the user's encryption key stored in the Network Interface Service (NIS), and then gives the decrypted key to the **keyserv** daemon to store away.

Normally, root's key is read from the file **/etc/.rootkey** when the daemon starts up. This is useful during power-failure reboots when no one is around to type a password, yet you still want the secure network services to operate normally.

OPTIONS

–d Prohibit the use of the default key. If this is used then every machine and user should have a publickey. New publickeys cannot be created if you do not already have a key. This can be done globally for an entire domain by deleting the **nobody** entry from **/etc/publickey** on the NIS master. See **chkey**(1)

–k Remember keylogins across machine reboots. This is only needed if **at**(1) is used to schedule jobs that require secure RPC. Use of this option is not recommended.

–n Do not read root's key from **/etc/.rootkey**. Instead, prompt the user for the password to decrypt **root** 's key stored in the NIS service and then store the decrypted key in **/etc/.rootkey** for future use. This option is useful if the **/etc/.rootkey** file ever gets out of date or corrupted.

FILES

/etc/.rootkey **/etc/keystore**

SEE ALSO

login(1), **keylogin**(1), **keylogout**(1), **publickey**(5)

NOTES

The Network Information Service (NIS) was formerly known as Sun Yellow Pages (YP). The functionality of the two remains the same; only the name has changed.

NAME

kgmon – generate a dump of the operating system's profile buffers

SYNOPSIS

/usr/etc/kgmon [**–bhpr**] [*filesystem*] [*memory*]

DESCRIPTION

kgmon is a tool used when profiling the operating system. When no arguments are supplied, **kgmon** indicates the state of operating system profiling as running, off, or not configured (see **config**(8)). If the **–p** flag is specified, **kgmon** extracts profile data from the operating system and produces a **gmon.out** file suitable for later analysis by **gprof**(1).

OPTIONS

–b Resume the collection of profile data.

–h Stop the collection of profile data.

–p Dump the contents of the profile buffers into a **gmon.out** file.

–r Reset all the profile buffers. If the **–p** flag is also specified, the **gmon.out** file is generated before the buffers are reset.

If neither **–b** nor **–h** is specified, the state of profiling collection remains unchanged. For example, if the **–p** flag is specified and profile data is being collected, profiling is momentarily suspended, the operating system profile buffers are dumped, and profiling is immediately resumed.

FILES

/vmunix	the default system
/dev/kmem	the default memory
gmon.out	

SEE ALSO

gprof(1), **config**(8)

DIAGNOSTICS

Users with only read permission on **/dev/kmem** cannot change the state of profiling collection. They can get a **gmon.out** file with the warning that the data may be inconsistent if profiling is in progress.

NAME

ldconfig – link-editor configuration

SYNOPSIS

/usr/etc/ldconfig [*directory* ...]

DESCRIPTION

ldconfig is used to configure a performance-enhancing cache for the run-time link-editor, **ld.so**. It is run from **/etc/rc.local** and periodically via **cron** to avoid linking with stale libraries. It should be also be run manually when a new shared object (e.g., a shared library) is installed on the system.

When invoked with no arguments, a default set of directories are built into the cache – these are the directories searched by default by the link editors. Additional directories may be specified on the command line.

FILES

/etc/ld.so.cache	holds the cached data.

SEE ALSO

ld(1)

NAME

link, unlink – exercise link and unlink system calls

SYNOPSIS

/usr/etc/link *filename1 filename2*

/usr/etc/unlink *filename*

AVAILABILITY

This command is available with the *System V* software installation option. Refer to *Installing SunOS 4.1* for information on how to install optional software.

DESCRIPTION

link and **unlink** perform their respective system calls on their arguments, abandoning all error checking.

SEE ALSO

rm(1), **link**(2V), **unlink**(2V)

WARNINGS

Only the super-user can unlink a directory, in which case the files it contains are lost. The files can, however, be recovered from the file system's **lost+found** directory after performing an **fsck**.

If you have write permission on the directory in which *filename* resides, **unlink** removes that file without warning, regardless of its ownership.

NAME

lockd, rpc.lockd – network lock daemon

SYNOPSIS

/usr/etc/rpc.lockd [**–g** *graceperiod*] [**–t** *timeout*]

DESCRIPTION

lockd processes lock requests that are either sent locally by the kernel or remotely by another lock daemon. **lockd** forwards lock requests for remote data to the server site's lock daemon through the **rpc**(3N) **xdr**(3N) in **lockd**(8C) package. **lockd** then requests the status monitor daemon, **statd**(8C), for monitor service. The reply to the lock request will not be sent to the kernel until the status daemon and the server site's lock daemon have replied.

If either the status monitor or server site's lock daemon is unavailable, the reply to a lock request for remote data is delayed until all daemons become available.

When a server recovers, it waits for a grace period for all client site lock daemons to submit reclaim requests. Client site lock daemons, on the other hand, are notified by the status daemon of the server recovery and promptly resubmit previously granted lock requests. If **lockd** fails to secure a previously granted lock at the server site, it sends SIGLOST to a process.

OPTIONS

–t *timeout*	Use *timeout* (seconds) as the interval instead of the default value (15 seconds) to retransmit lock request to the remote server.
–g *graceperiod*	Use *graceperiod* (**seconds**) as the grace period duration instead of the default value (45 seconds).

SEE ALSO

fcntl(2V), **lockf**(3), **signal**(3V), **statd**(8C)

NAME

logintool – graphic login interface

AVAILABILITY

Available only on Sun 386i systems running a SunOS 4.0*x* release or earlier. Not a SunOS 4.1 release feature.

DESCRIPTION

logintool is started by **getty**(8) to display a full screen window for logging in. It cannot be run from the shell. It is more attractive than the traditional '**login:** ' prompt, and also provides help for the person without a username and information about the workstation.

logintool is normally invoked on the console by **getty**(8), and works only on a frame buffer.

If the **newlogin** policy in the **policies** Network Interface Service (NIS) map is set to **unrestricted**, then **logintool** may create new user accounts in the NIS service. The account resides on the local system if it is diskful, or on the system's boot server if the local system is diskless.

FILES

/usr/share/lib/ez/login

SEE ALSO

getty(8)

NOTES

The Network Information Service (NIS) was formerly known as Sun Yellow Pages (YP). The functionality of the two remains the same; only the name has changed.

NAME

lpc – line printer control program

SYNOPSIS

/usr/etc/lpc [*command* [*parameter*...]]

DESCRIPTION

lpc controls the operation of the printer, or of multiple printers, as described in the **/etc/printcap** database. **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.

USAGE

Commands

Commands may be abbreviated to an unambiguous substring. Note: the *printer* parameter is specified just by the name of the printer (as **lw**), not as you would specify it to **lpr**(1) or **lpq**(1) (not as **–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 super-user.

clean [**all** | [*printer*...]]

Remove all files with names beginning with **cf**, **tf**, or **df** from the specified printer queue(s) on the local machine. The **clean** command can only be used by the super-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 super-user.

down [**all** | [*printer*...]] [*message*]

Turn the specified printer queue off, disable printing and put *message* in the printer status file. The message doesn't need to be quoted, the remaining arguments are treated like **echo**(1V). 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 super-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. **lpq**(1) reports that there is no daemon present when this condition occurs. 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 super-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 super-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 super-user.

up [**all** | [*printer* ...]] Enable everything and start a new printer daemon. Undoes the effects of **down**.

FILES

/etc/printcap	printer description file
/var/spool/*	spool directories
/var/spool/*/lock	lock file for queue control

SEE ALSO

lpq(1), **lpr**(1), **lprm**(1), **printcap**(5), **lpd**(8)

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 super-user.

NAME

lpd – printer daemon

SYNOPSIS

/usr/lib/lpd [**–l**] [**–L** *logfile*] [*port#*]

DESCRIPTION

lpd is the line printer daemon (spool area handler). It is usually invoked at boot time from the **rc**(8) script, making a single pass through the **printcap**(5) file to find out about the existing printers and printing any files left after a crash. It then accepts requests to print files in a queue, transfer files to a spooling area, display a queue's status, or remove jobs from a queue. In each case, it forks a child process for each request, and continues to listen for subsequent requests.

The Internet port number used to communicate with other processes is usually obtained with **getservent**(3N), but can be specified with the *port#* argument.

If a file cannot be opened, an error message is logged using the LOG_LPR facility of **syslog**(3). **lpd** will try up to 20 times to reopen a file it expects to be there, after which it proceeds to the next file or job.

OPTIONS

–l Log valid requests received from the network. This can be useful for debugging purposes.

–L *logfile*
Change the file used for writing error conditions to *logfile*. The default is to report a message using the **syslog**(3) facility.

OPERATION

Access Control

Access control is provided by two means. First, all requests must come from one of the machines listed in either the file **/etc/hosts.equiv** or **/etc/hosts.lpd**. (This latter file is in **hosts.equiv**(5) format.) Second, if the **rs** capability is specified in the **printcap** entry, **lpr**(1) requests are only be honored for users with accounts on the printer host.

Lock File

The **lock** file in each spool directory is used to prevent multiple daemons from becoming active, and to store information about the daemon process for **lpr**(1), **lpq**(1), and **lprm**(1).

lpd uses **flock**(2) to provide exclusive access to the lock file and to prevent multiple daemons from becoming active simultaneously. If the daemon should be killed or die unexpectedly, the lock file need not be removed. The lock file is kept in a readable ASCII form and contains two lines. The first is the process id of the daemon and the second is the control file name of the current job being printed. The second line is updated to reflect the current status of **lpd** for the programs **lpq**(1) and **lprm**(1).

Control Files

After the daemon has successfully set the lock, it scans the directory for files beginning with **cf**. Lines in each *cf* file specify files to be printed or non-printing actions to be performed. Each such line begins with a key character that indicates what to do with the remainder of the line.

J Job name to print on the burst page.
C Classification line on the burst page.
L Literal. This line contains identification information from the password file, and causes a burst page to be printed.
T Title string for page headings printed by **pr**(1V).
H Hostname of the machine where **lpr**(1) was invoked.
P Person. Login name of the person who invoked **lpr**(1). This is used to verify ownership by **lprm**(1).
M Send mail to the specified user when the current print job completes.
f Formatted File, the name of a file to print that is already formatted.
l Like **f**, but passes control characters along, and does not make page breaks.
p Name of a file to print using **pr**(1V) as a filter.
t Troff File. The file contains **troff**(1) output (cat phototypesetter commands).

n	Ditroff File. The file contains device independent troff output.
d	DVI File. The file contains TEX output (DVI format from Stanford).
g	Graph File. The file contains data produced by **plot**(3X).
c	Cifplot File. The file contains data produced by *cifplot*.
v	The file contains a raster image.
r	The file contains text data with FORTRAN carriage control characters.
1	Troff Font R. The name of a font file to use instead of the default.
2	Troff Font I. The name of the font file to use instead of the default.
3	Troff Font B. The name of the font file to use instead of the default.
4	Troff Font S. The name of the font file to use instead of the default.
W	Width. Changes the page width (in characters) used by **pr**(1V) and the text filters.
I	Indent. Specify the number of characters by which to indent the output.
U	Unlink. The name of file to remove upon completion of printing.
N	Filename. The name of the file being printed, or a blank for the standard input (when **lpr**(1) is invoked in a pipeline).

Data Files

When a file is spooled for printing, the contents are copied into a data file in the spool directory. Data file names begin with **df**. When **lpr** is called with the **-s** option, the control files contain a symbolic link to the actual file, and no data files are created.

Minfree File

The file *minfree* in each spool directory contains the number of kilobytes to leave free so that the line printer queue won't completely fill the disk.

FILES

/etc/printcap	printer description file
/var/spool/*	spool directories
/var/spool/*/minfree	minimum free space to leave
/dev/lp*	line printer devices
/dev/printer	socket for local requests
/etc/hosts.equiv	hosts allowed equivalent host access
/etc/hosts.lpd	hosts allowed printer access only

SEE ALSO

lpq(1), **lpr**(1), **lprm**(1), **hosts**(5), **hosts.equiv**(5), **printcap**(5), **lpc**(8), **pac**(8)

NAME

mailstats – print statistics collected by sendmail

SYNOPSIS

/usr/etc/mailstats [*filename*]

DESCRIPTION

mailstats prints out the statistics collected by the **sendmail** program on mailer usage. These statistics are collected if the file indicated by the S configuration option of **sendmail** exists. The **mailstats** program first prints the time that the statistics file was created and the last time it was modified. It will then print a table with one row for each mailer specified in the configuration file. The first column is the mailer number, followed by the symbolic name of the mailer. The next two columns refer to the number of messages received by *sendmail*, and the last two columns refer to messages sent by *sendmail*. The number of messages and their total size (in 1024 byte units) is given. No numbers are printed if no messages were sent (or received) for any mailer.

You might want to add an entry to **/var/spool/cron/crontab/root** to reinitialize the statistics file once a night. Copy **/dev/null** into the statistics file or otherwise truncate it to reset the counters.

FILES

/etc/sendmail.st	default statistics file
/etc/sendmail.cf	sendmail configuration file
/var/spool/cron/crontab/root	
/dev/null	

SEE ALSO

sendmail(8)

BUGS

Mailstats should read the configuration file instead of having a hard-wired table mapping mailer numbers to names.

NAME

makedbm – make a NIS ndbm file

SYNOPSIS

/usr/etc/yp/makedbm [**–b**] [**–l**] [**–s**] [**–i** *yp_input_file*] [**–o** *yp_output_name*] [**–d** *yp_domain_name*] [**–m** *yp_master_name*] *infile outfile*

makedbm [**–u** *dbmfilename*]

DESCRIPTION

makedbm takes *infile* and converts it to a pair of files in **ndbm**(3) format, namely *outfile*.**pag** and *outfile*.**dir**. Each line of the input file is converted to a single **dbm** record. All characters up to the first TAB or SPACE form the key, and the rest of the line is the data. If a line ends with '\', then the data for that record is continued on to the next line. It is left for the clients of the Network Interface Service (NIS) to interpret #; **makedbm** does not itself treat it as a comment character. *infile* can be '–', in which case the standard input is read.

makedbm is meant to be used in generating **dbm** files for the NIS service, and it generates a special entry with the key *yp_last_modified*, which is the date of *infile* (or the current time, if *infile* is '–').

OPTIONS

–b Interdomain. Propagate a map to all servers using the interdomain name server **named**(8C).

–l Lowercase. Convert the keys of the given map to lower case, so that host name matches, for example, can work independent of upper or lower case distinctions.

–s Secure map. Accept connections from secure NIS networks only.

–i *yp_input_file*
Create a special entry with the key *yp_input_file*.

–o *yp_output_name*
Create a special entry with the key *yp_output_name*.

–d *yp_domain_name*
Create a special entry with the key *yp_domain_name*.

–m *yp_master_name*
Create a special entry with the key *yp_master_name*. If no master host name is specified, *yp_master_name* will be set to the local host name.

–u *dbmfilename*
Undo a **dbm** file. That is, print out a **dbm** file one entry per line, with a single space separating keys from values.

EXAMPLE

It is easy to write shell scripts to convert standard files such as **/etc/passwd** to the key value form used by **makedbm**. For example:

```
#!/bin/awk -f
BEGIN { FS = ":"; OFS = "\t"; }
{ print $1, $0 }
```

takes the **/etc/passwd** file and converts it to a form that can be read by **makedbm** to make the NIS file **passwd.byname**. That is, the key is a username, and the value is the remaining line in the **/etc/passwd** file.

SEE ALSO

yppasswd(1), **ndbm**(3), **named**(8C)

NOTES

The Network Information Service (NIS) was formerly known as Sun Yellow Pages (YP). The functionality of the two remains the same; only the name has changed.

NAME

makedev, MAKEDEV – make system special files

SYNOPSIS

/dev/MAKEDEV *device-name* ...

DESCRIPTION

MAKEDEV is a shell script normally used to install special files. It resides in the /dev directory, as this is the normal location of special files. Arguments to **MAKEDEV** are usually of the form *device-name?* where *device-name* is one of the supported devices listed in section 4 of the manual and '*?*' is a logical unit number (0-9). A few special arguments create assorted collections of devices and are listed below.

std Create the *standard* devices for the system; for example, **/dev/console**, **/dev/tty**.

local Create those devices specific to the local site. This request runs the shell file **/dev/MAKEDEV.local**. Site specific commands, such as those used to setup dialup lines as "ttyd?" should be included in this file.

Since all devices are created using **mknod**(8), this shell script is useful only to the super-user.

FILES

/dev/console /dev/MAKEDEV.local /dev/tty

SEE ALSO

intro(4), **config**(8), **mknod**(8)

DIAGNOSTICS

Either self-explanatory, or generated by one of the programs called from the script. Use **sh –x MAKEDEV** in case of trouble.

NAME

makekey – generate encryption key

SYNOPSIS

/usr/lib/makekey

DESCRIPTION

makekey improves the usefulness of encryption schemes depending on a key by increasing the amount of time required to search the key space. It reads 10 bytes from its standard input, and writes 13 bytes on its standard output. The output depends on the input in a way intended to be difficult to compute (that is, to require a substantial fraction of a second).

The first eight input bytes (the *input key*) can be arbitrary ASCII characters. The last two (the *salt*) are best chosen from the set of digits, upper- and lower-case letters, and '.' and '/'. The salt characters are repeated as the first two characters of the output. The remaining 11 output characters are chosen from the same set as the salt and constitute the *output key*.

The transformation performed is essentially the following: the salt is used to select one of 4096 cryptographic machines all based on the National Bureau of Standards DES algorithm, but modified in 4096 different ways. Using the input key as key, a constant string is fed into the machine and recirculated a number of times. The 64 bits that come out are distributed into the 66 useful key bits in the result.

makekey is intended for programs that perform encryption (for instance, **ed**(1) and **crypt**(1)). Usually makekey's input and output will be pipes.

SEE ALSO

crypt(1), **ed**(1)

NAME

mc68881version – print the MC68881 mask number and approximate clock rate

SYNOPSIS

/usr/etc/mc68881version

AVAILABILITY

Sun-2, Sun-3, and Sun-4 systems only.

DESCRIPTION

mc68881version determines whether an MC68881 or MC68882 floating-point coprocessor is available, and if so, determines its apparent mask number and approximate clock rate and prints them on the standard output. The reported clock rate is derived by timing floating-point operations with **getrusage**(2) and is thus somewhat variable; best results may be obtained in single-user mode. The same applies to the differentiation between MC68881 and MC68882 ; these can be distinguished in user mode only by timing tests.

SEE ALSO

getrusage(2)

NAME

mconnect – connect to SMTP mail server socket

SYNOPSIS

/usr/etc/mconnect [**–p** *port*] [**–r**] [*hostname*]

DESCRIPTION

mconnect opens a connection to the mail server on a given host, so that it can be tested independently of all other mail software. If no host is given, the connection is made to the local host. Servers expect to speak the Simple Mail Transfer Protocol (SMTP) on this connection. Exit by typing the **quit** command. Typing EOF will send an end of file to the server. An interrupt closes the connection immediately and exits.

OPTIONS

–p *port* Specify the port number instead of the default SMTP port (number 25) as the next argument.

–r "Raw" mode: disable the default line buffering and input handling. This gives you a similar effect as **telnet** to port number 25, not very useful.

FILES

/usr/lib/sendmail.hf help file for SMTP commands

SEE ALSO

sendmail(8)

Postel, Jonathan B *Simple Mail Transfer Protocol*, RFC821 August 1982, SRI Network Information Center

NAME

mkfile – create a file

SYNOPSIS

mkfile [**–nv**] *size*[**k**|**b**|**m**] *filename* ...

DESCRIPTION

mkfile creates one or more files that are suitable for use as NFS-mounted swap areas, or as local swap areas. The sticky bit is set, and the file is padded with zeroes by default. The default *size* is in bytes, but it can be flagged as kilobytes, blocks, or megabytes, with the **k**, **b**, or **m** suffixes, respectively.

OPTIONS

–n Create an empty *filename*. The size is noted, but disk blocks aren't allocated until data is written to them.

–v Verbose. Report the names and sizes of created files.

SEE ALSO

swapon(2), **fstab**(5), **swapon**(8)

NAME

mkfs – construct a file system

SYNOPSIS

/usr/etc/mkfs [–N] *special size* [*nsect*] [*ntrack*] [*blksize*] [*fragsize*] [*ncpg*] [*minfree*] [*rps*] [*nbpi*] [*opt*] [*apc*] [*rot*] [*nrpos*]

DESCRIPTION

Note: file systems are normally created with the **newfs**(8) command.

mkfs constructs a file system by writing on the special file *special* unless the –N flag has been specified. *special* must be specified as a raw device and disk partition. For example, to create a file system on **sd0**, specify **/dev/rsd0[a-h]**, where **a-h** is the disk partition.

The numeric *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**(8)). The number of inodes is calculated as a function of the file system size. No boot program is initialized by **mkfs** (see **newfs**(8)).

You must be super-user to use this command.

OPTIONS

–N Print out the file system parameters without actually creating the file system.

The following arguments allow fine tune control over the parameters of the file system.

nsect The number of sectors per track on the disk. The default is **32**.

ntrack The number of tracks per cylinder on the disk. The default is **16**.

blksize The primary block size for files on the file system. It must be a power of two, currently selected from **4096** or **8192** (the default).

fragsize 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 **8192**. The default is **1024**.

ncpg The number of disk cylinders per cylinder group. The default is **16**.

minfree The minimum percentage of free disk space allowed. Once the file system capacity reaches this threshold, only the super-user is allowed to allocate disk blocks. The default value is **10%**.

rps The rotational speed of the disk, in revolutions per second. The default is **60**.

nbpi The number of bytes for which one inode block is allocated. This parameter is currently set at one inode block for every 2048 bytes.

opt Space or time optimization preference; **s** specifies optimization for space, **t** specifies optimization for time. The default is **t**.

apc The number of alternates per cylinder (SCSI devices only). The default is **0**.

rot 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.

nrpos The number of distinguished rotational positions. The default is 8.

Users with special demands for their file systems are referred to the paper cited below for a discussion of the tradeoffs in using different configurations.

SEE ALSO

dir(5), **fs**(5), **fsck**(8), **newfs**(8), **tunefs**(8)

System and Network Administration
McKusick, Joy, Leffler; *A Fast File System for UNIX*

NOTES

newfs(8) is preferred for most routine uses.

NAME

mknod – build special file

SYNOPSIS

/usr/etc/mknod *filename* [**c**] [**b**] *major minor*

/usr/etc/mknod *filename* **p**

DESCRIPTION

mknod makes a special file. The first argument is the *filename* of the entry. In the first form, the second argument is **b** if the special file is block-type (disks, tape) or **c** if it is character-type (other devices). The last two arguments are numbers specifying the *major* device type and the *minor* device (for example, unit, drive, or line number). Only the super-user is permitted to invoke this form of the **mknod** command.

In the second form, **mknod** makes a named pipe (FIFO).

The first form of **mknod** is only for use by system configuration people. Normally you should use **/dev/MAKEDEV** instead when making special files.

SEE ALSO

mknod(2V), **makedev**(8)

NAME

mkproto – construct a prototype file system

SYNOPSIS

/usr/etc/mkproto *special proto*

DESCRIPTION

mkproto is used to bootstrap a new file system. First a new file system is created using **newfs**(8). **mkproto** is then used to copy files from the old file system into the new file system according to the directions found in the prototype file **proto**. The prototype file contains tokens separated by SPACE or NEWLINE characters. The first tokens comprise the specification for the root directory. File specifications consist of tokens giving the mode, the user ID, the group ID, and the initial contents of the file. The syntax of the contents field depends on the mode.

The mode token for a file is a 6 character string. The first character specifies the type of the file. (The characters **–bcd** specify regular, block special, character special and directory files respectively.) The second character of the type is either **u** or '–' to specify set-user-id mode or not. The third is **g** or '–' for the set-group-id mode. The rest of the mode is a three digit octal number giving the owner, group, and other read, write, execute permissions, see **chmod**(1V).

Two decimal number tokens come after the mode; they specify the user and group ID's of the owner of the file.

If the file is a regular file, the next token is a pathname whence the contents and size are copied.

If the file is a block or character special file, two decimal number tokens follow which give the major and minor device numbers.

If the file is a directory, **mkproto** makes the entries '**.**' and '**..**' and then reads a list of names and (recursively) file specifications for the entries in the directory. The scan is terminated with the token $.

A sample prototype specification follows:

```
d--777 3 1
usr     d--777 3 1
        sh      ---755 3 1 /usr/bin/sh
        ken     d--755 6 1
                $
        b0      b--644 3 1 0 0
        c0      c--644 3 1 0 0
        $
$
```

SEE ALSO

chmod(1V), **fs**(5), **dir**(5), **fsck**(8), **newfs**(8)

BUGS

There should be some way to specify links.

There should be some way to specify bad blocks.

mkproto can only be run on virgin file systems. It should be possible to copy files into existent file systems.

NAME

modload – load a module

SYNOPSIS

modload *filename* [**–conf** *config_file*] [**–entry** *entry_point*] [**–exec** *exec_file*] [**–o** *output_file*] [**–nolink**] [**–A** *vmunix_file*]

DESCRIPTION

modload loads a loadable module into a running system. The input file *filename* is an object file (**.o** file).

OPTIONS

–conf *config_file*

Use this configuration file to configure the loadable driver being loaded. The commands in this file are the same as those that the **config**(8) program recognizes. There are two additional commands, **blockmajor** and **charmajor**, shown in the configuration file example below.

–entry *entry_point*

This is the module entry point. This is passed by **modload** to **ld**(1) when the module is linked. The default module entry point name is '*xxx* **init**'.

–exec *exec_file*

This is the name of a shell script or executable image file that is executed if the module is successfully loaded. It is always passed the module id and module type as the first two arguments. For loadable drivers, the third and fourth arguments are the block major and character major numbers respectively. For a loadable system call, the third argument is the system call number.

–o *output_file*

This is the name of the output file that is produced by the linker. If this option is omitted, then the output file name is *filename*> without the '**.o**'.

–nolink This option can be used if **modload** has already been issued once and the output file already exists. One must take care that neither the kernel nor the module have changed.

–A *vmunix_file*

This is the file that is passed to the linker to resolve module references to kernel symbols. The default is **/vmunix**. The symbol file must be for the currently running kernel or the module is likely to crash the system.

EXAMPLES

```
controller      fdc0 at atmem csr 0x001000 irq 6 priority 3
controller      fdc2 at atmem csr 0x002000 irq 5 priority 2
disk            fd0 at fdc0 drive 0
disk            fd0 at fdc0 drive 1
disk            fd0 at fdc0 drive 2
device          fd0 at fdc2 drive 0 csr 0x003000 irq 4 priority 2
disk            fd0 at fdc2 drive 1
blockmajor 51
charmajor 52
```

SEE ALSO

ld(1), **modunload**(8), **modstat**(8)

NAME

modstat – display status of loadable modules

SYNOPSIS

modstat [**–id** *module_id*]

DESCRIPTION

modstat displays the status of the loaded modules.

OPTIONS

–id *module_id*

Display the status of only this module.

SEE ALSO

modload(8), **modunload**(8)

NAME

modunload – unload a module

SYNOPSIS

modunload –id *module_id* [**–exec exec_file**]

DESCRIPTION

modunload unloads a loadable module from a running system. The *module_id* is the ID of the module as shown by **modstat**(8).

OPTIONS

–exec *exec_file*

This is the name of a shell script or executable image file that will be executed before the module is unloaded. It is always passed the module ID and module type as the first two arguments. For loadable drivers, the third and fourth arguments are the block major and character major numbers respectively. For a loadable system call, the third argument is the system call number.

SEE ALSO

modload(8), **modstat**(8)

NAME

monitor – system ROM monitor

SYNOPSIS

L1–A

BREAK

DESCRIPTION

The CPU board of the Sun workstation contains an EPROM (or set of EPROMs), called the *monitor*, that controls the system during startup. The monitor tests the system before attempting to boot the operating system. If you interrupt the boot procedure by holding down **L1** while typing **a** or **A** on the workstation keyboard (or **BREAK** if the console is a dumb terminal) the monitor issues the prompt:

>

and accepts commands interactively.

USAGE

Modes

The monitor supports three security modes (non-secure, command secure, and fully secure) and an authentication password. Access to monitor commands is controlled by these security modes. In **non-secure** mode all monitor commands are allowed. In **command secure** mode, only the **b**(boot) command with no arguments and the **c**(continue) command with no arguments may be entered without supplying the authentication password. In **fully secure** mode, only the **c**(continue) command with no arguments may be entered without supplying the authentication password. Note: The system will not auto-reboot in fully secure mode. The authentication password must be entered before booting will take place.

Commands

+|– Increment or decrement the current address and display the contents of the new location.

^C *source destination n*
(caret-C) Copy, byte-by-byte a block of length *n* from the *source* address to the *destination* address.

^I *program* (caret-I) Display the compilation date and location of *program*.

^T *virtual_address*
(caret-T) Display the physical address to which *virtual_address* is mapped.

a [*n*] [*action*]... (Sun-2 and Sun-3 systems only)
Open **A**–register (cpu address register) *n*, and perform indicated actions. The number *n* can be any value from **0** to **7**, inclusive. The default value is **0**. A hexadecimal *action* argument assigns the value you supply to the register *n*. A non-hex *action* terminates command input.

b [**!**] [*device* [(*c*,*u*,*p*)]] [*pathname*] [*arguments_list*]
b[?] Reset appropriate parts of the system and bootstrap a program. A '**!**' (preceding the *device* argument) prevents the system reset from occurring. Programs can be loaded from various devices (such as a disk, tape or Ethernet). '**b**' with no arguments will cause a default boot, either from a disk, or from an Ethernet controller. '**b?**' displays all boot devices and their *device* arguments, where *device* is one of:

ie	Intel Ethernet
le	Lance Ethernet (Sun-2, Sun-3, Sun-4 systems only)
sd	SCSI disk
st	SCSI 1/4" tape
mt	Tape Master 9-track 1/2" tape (Sun-2, Sun-3, Sun-4 systems only)
xd	Xylogics 7053 disk (Sun-2, Sun-3, Sun-4 systems only)
xt	Xylogics 1/2" tape (Sun-2, Sun-3, Sun-4 systems only)
xy	Xylogics 440/450 disk (Sun-2, Sun-3, Sun-4 systems only)
fd	Diskette (Sun386i system only)

c A controller number (**0** if only one controller),

u A unit number (**0** if only one driver), and

p A partition.

pathname A pathname for a program such as **/stand/diag**. **/vmunix** is the default.

arguments_list
A list of up to seven arguments to pass to the program being booted.

c [*virtual_address*]
Resume execution of a program. When given, *virtual_address* is the address at which execution will resume. The default is the current PC (EIP on Sun386i systems). Registers are restored to the values shown by the **a**, **d**, and **r** commands (for Sun-2 and Sun-3 systems), or by the **d** and **r** commands (for Sun-4 systems), or by the **d** command (for Sun386i systems).

d [*window_number*] (Sun-4 systems only)
Display (dump) the state of the processor. The processor state is observable only after:

- An unexpected trap was encountered.
- A user program dropped into the monitor (by calling *abortent*).
- The user manually entered the monitor by typing **L1–A** or **BREAK**.

The display consists of the following:

- The special registers: PSR, PC, nPC, TBR, WIM and Y
- Eight global registers, and
- 24 window registers (8 *in*, 8 *local*, and 8 *out*), corresponding to one of the 7 available windows. If a Floating-Point Unit is on board, its status register along with its 32 floating-point registers are also shown.

window_number
Display the indicated *window_number*, which can be any value between 0 and 6, inclusive. If no window is specified and the PSR's current window pointer contains a valid window number, registers from the window that was active just prior to entry into the monitor are displayed. Otherwise, registers from window 0 are displayed.

d (Sun386i systems only)
Display (dump) the state of the processor. This display consists of the registers, listed below:

Processor Registers:	EAX, ECX, EDX, ESI, EDI, ESP, EBP, EFLAGS, EIP
Segment Registers:	ES, CS, SS, DS, FS, GS
Memory Management Registers:	GDTR, LDTR, IDTR, TR
Control Registers:	CR0, CR2, CR3
Debug Registers:	DR0, DR1 , DR2 , DR3, DR6, DR7
Test Registers:	TR6, TR7

The processor's state is observable only after an unexpected trap, a user program has "dropped" into the monitor (by calling monitor function abortentor) or the user has manually "broken " into the monitor (by typing **L1–A** on the Workstation console, or **BREAK** on the dumb terminal's keyboard.

d [*n*] [*action*]... (Sun-2 and Sun-3 systems only)
Open **D**–register (cpu data register) *n*, and perform indicated actions. The number *n* can be any value from **0** to **7**, inclusive. The default is **0**. See the **a** command for a description of *action*.

e [*virtual_address*] [*action*] ...

Open the 16 bit word at *virtual_address* (default zero). On Sun-2, Sun-3, and Sun-4 systems, the address is interpreted in the address space defined by the **s** command. See the **a** command for a description of *action*.

f *virtual_address1 virtual_address2 pattern* [*size*] (Sun-3 and Sun-4 systems only)

Fill the bytes, words or long words from *virtual_address1* (lower) to *virtual_address2* (higher) with the constant, *pattern*. The *size* argument can take one of the following values

b byte format (the default)
w word format
l long word format

For example, the following command fills the address block from 0x1000 to 0x2000 with the word pattern, 0xABCD:

f 1000 2000 ABCD W

g [*vector*] [*argument*]
g [*virtual_address*] [*argument*]

Goto (jump to) a predetermined or default routine (first form), or to a user-specified routine (second form). The value of *argument* is passed to the routine. If the *vector* or *virtual_address* argument is omitted, the value in the PC is used as the address to jump to.

To set up a predetermined routine to jump to, a user program must, prior to executing the monitor's **g** command, set the variable ***romp->v_vector_cmd** to be equal to the virtual address of the desired routine. Predetermined routines need not necessarily return control to the monitor.

The default routine, defined by the monitor, prints the user-supplied *vector* according to the format supplied in *argument*. This format can be one of:

%x hexadecimal
%d decimal

g0 (Sun-2, Sun-3, and Sun-4 only)

When the monitor is running as a result of the system being interrupted, force a panic and produce a crash dump.

g4

When the monitor is running as a result of the system being interrupted, force a kernel stack trace.

h (Sun-3 and Sun-4 and Sun386i systems)

Display the help menu for monitor commands and their descriptions. To return to the monitor's basic command level, press ESCAPE or **q** before pressing RETURN.

i [*cache_data_offset*] [*action*] ... (Sun-3/200 series and Sun-4 systems only)

Modify cache data RAM command. Display and/or modify one or more of the
Modify cache data RAM command. Display and/or modify one or more of the cache data addresses. See the **a** command for a description of *action*.

j [*cache_tag_offset*] [*action*] ... (Sun-3/200 series and Sun-4 systems only)

Modify cache tag RAM command. Display and/or modify the contents of one or more of the cache tag addresses. See the **a** command for a description of *action*.

k [*reset_level*]

Reset the system. If *reset_level* is:

0 CPU reset only (Sun-2 and Sun-3 systems). Reset VMEbus, interrupt registers, video monitor (Sun-4 systems). This is the default. Reset video (Sun386i systems).
1 Software reset.

2 Power-on reset. Resets and clears the memory. Runs the EPROM-based diagnostic self test, which can take several minutes, depending upon how much memory is being tested.

kb Display the system banner.

l [*virtual_address*] [*action*] ...
Open the long word (32 bit) at memory address *virtual_address* (default zero). On Sun-2, Sun-3 and Sun-4 systems, the address is interpreted in the address space defined by the **s** command (below). See the **a** command for a description of *action*.

m [*virtual_address*] [*action*] ...
Open the segment map entry that maps *virtual_address* (default zero). On Sun-2, Sun-3 and Sun-4 systems, the address is interpreted in the address space defined by the **s** command. Not supported on Sun386i. See the **a** command for a description of *action*.

nd (Sun386i systems only)
ne
ni Disable, enable, or invalidate the cache, respectively

o [*virtual_address*] [action*]* ...
Open the byte location specified by
virtual_address (default zero). On Sun-2, Sun-3 and Sun-4 systems, the address is interpreted in the address space defined by the **s** command. See the **a** command for a description of *action*.

p [*virtual_address*] [*action*]...
Open the page map entry that maps *virtual_address* (default zero) in the address space defined by the **s** command. See the **a** command for a description of *action*.

p [*port_address*] [[*nonhex_char* [*hex_value*] | *hex_value*] ...] (Sun386i systems only)
Display or modify the contents of one or more port I/O addresses in byte mode. Each port address is treated as a 8-bit unit. The optional *port_address*, argument, which is a 16-bit quantity, specifies the initial port I/O address. See the **e** command for argument descriptions.

q [*eeprom_offset*] [*action*]... (Sun-3 and Sun-4 systems only)
Open the EEPROM *eeprom_offset* (default zero) in the EEPROM address space. All addresses are referenced from the beginning or base of the EEPROM in physical address space, and a limit check is performed to insure that no address beyond the EEPROM physical space is accessed. On Sun386i systems, open the NVRAM *nvram_offset* (default zero). This command is used to display or modify configuration parameters, such as: the amount of memory to test during self test, whether to display a standard or custom banner, if a serial port (A or B) is to be the system console, etc. See the **a** command for a description of *action*.

r [*reg_name*] [[*nonhex_char* [*hex_value*] | *hex_value*] ...] (Sun386i systems only)
Display or modify one or more of the processor registers. If *reg_name* is specified (2 or 3 characters from the above list), that register is displayed first. The default is **EAX**. See note on register availability under the command **d** (for Sun386i systems). See the **e** command for argument descriptions.

s [*step_count*] (Sun386i systems only)
Single step the execution of the interrupted program. The *step_count* argument specifies the number of single steps to execute before displaying the monitor prompt. The default is 1.

r [*register_number*] [*action*]... (Sun-2 and Sun-3 systems only)
Display and/or modify the register indicated. *register_number* can be one of:

CA 68020 Cache Address Register
CC 68020 Cache Control Register
CX 68020 System and User Context

DF Destination Function code
IS 68020 Interrupt Stack Pointer
MS 68020 Master Stack Pointer
PC Program Counter
SC 68010 System Context
SF Source Function code
SR Status Register
SS 68010 Supervisor Stack Pointer
UC 68010 User Context
US User Stack Pointer
VB Vector Base

Alterations to these registers (except **SC** and **UC**) do not take effect until the next **c** command is executed. See the **a** command for a description of *action*.

r [*register_number*] (Sun-4 systems only)
r [*register_type*]
r [*w window_number*]

Display and/or modify one or more of the IU or FPU registers.

A hexadecimal *register_number* can be one of:

0x00—0x0f window(0,i0)—window(0,i7), window(0,i0)—window(0,i7)
0x16—0x1f window(1,i0)—window(1,i7), window(1,i0)—window(1,i7)
0x20—0x2f window(2,i0)—window(2,i7), window(2,i0)—window(2,i7)
0x30—0x3f window(3,i0)—window(3,i7), window(3,i0)—window(3,i7)
0x40—0x4f window(4,i0)—window(4,i7), window(4,i0)—window(4,i7)
0x50—0x5f window(5,i0)—window(5,i7), window(5,i0)—window(5,i7)
0x60—0x6f window(6,i0)—window(6,i7), window(6,i0)—window(6,i7)
0x70—0x77 g0, g1, g2, g3, g4, g5, g6, g7
0x78—0x7d PSR, PC, nPC, WIM, TBR, Y
0x7e—0x9e FSR, f0—f31

Register numbers can only be displayed after an unexpected trap, a user program has entered the monitor using the *abortent* function, or the user has entered the monitor by manually typing **L1–A** or **BREAK**.

If a *register_type* is given, the first register of the indicated type is displayed. *register_type* can be one of:

f floating-point
g global
s special

If **w** and a *window_number* (**0—6**) are given, the first *in*-register within the indicated window is displayed. If *window_number* is omitted, the window that was active just prior to entering the monitor is used. If the PSR's current window pointer is invalid, window 0 is used.

s [*code*] (Sun-2 and Sun-3 systems only)

Set or query the address space to be used by subsequent memory access commands. *code* is one of:

0	undefined
1	user data space
2	user program space
3	user control space
4	undefined
5	supervisor data space
6	supervisor program space
7	supervisor control space

If *code* is omitted, **s** displays the current address space.

s [*asi*] (Sun-4 systems only)

Set or display the Address Space Identifier. With no argument, **s** displays the current Address Space Identifier. The *asi* value can be one of:

0x2	control space
0x3	segment table
0x4	Page table
0x8	user instruction
0x9	supervisor instruction
0xa	user data
0xb	supervisor data
0xc	flush segment
0xd	flush page
0xe	flush context
0xf	cache data

t [*program*] (Sun-3 systems only)

Trace the indicated standalone *program*. Works only with programs that do not affect interrupt vectors.

u [*echo*]
u [*port*] [*options*] [*baud_rate*]
u [**u**] [*virtual_address*]

With no arguments, display the current I/O device characteristics including: current input device, current output device, baud rates for serial ports A and B, an input-to-output echo indicator, and virtual addresses of mapped UART devices. With arguments, set or configure the current I/O device. With the **u** argument (**uu**...), set the I/O device to be the *virtual_address* of a UART device currently mapped.

echo Can be either **e** to enable input to be echoed to the output device, or **ne**, to indicate that input is not echoed.

port Assign the indicated *port* to be the current I/O device. *port* can be one of:

a	serial port A
b	serial port B (except on Sun386i systems)
k	the workstation keyboard
s	the workstation screen

baud_rate Any legal baud rate.

options can be any combination of:

i	input
o	output

u UART
e echo input to output
ne do not echo input
r reset indicated serial port (**a** and **b** ports only)

If either **a** or **b** is supplied, and no *options* are given, the serial port is assigned for both input and output. If **k** is supplied with no options, it is assigned for input only. If **s** is supplied with no options, it is assigned for output only.

v *virtual_address1 virtual_address2* [*size*] (Sun-3 and Sun-4 systems only)

Display the contents of *virtual_address1* (lower) *virtual_address2* (higher) in the format specified by *size*:

b byte format (the default)
w word format
l long word format

Enter return to pause for viewing; enter another return character to resume the display. To terminate the display at any time, press the space bar.

For example, the following command displays the contents of virtual address space from address 0x1000 to 0x2000 in word format:

v 1000 2000 W

w [*virtual_address*] [*argument*] (Sun-3 and Sun-4 systems only)

Set the execution vector to a predetermined or default routine. Pass *virtual_address* and *argument* to that routine.

To set up a predetermined routine to jump to, a user program must, prior to executing the monitor's **w** command, set the variable ***romp->v_vector_cmd** to be equal to the virtual address of the desired routine. Predetermined routines need not necessarily return control to the monitor.

The default routine, defined by the monitor, prints the user-supplied *vector* according to the format supplied in *argument*. This format can be one of:

%x hexadecimal
%d decimal

x (Sun-3 and Sun-4 systems only)

Display a menu of extended tests. These diagnostics permit additional testing of such things as the I/O port connectors, video memory, workstation memory and keyboard, and boot device paths.

y c *context_number* (Sun-4 systems only)
y p|**s** *context_number virtual_address*

Flush the indicated context, context page, or context segment.

c flush context *context_number*
p flush the page beginning at *virtual_address* within context *context_number*
s flush the segment beginning at *virtual_address* within context *context_number*

z [*number*] [*breakpoint_virtual_address* [*type*] [*len*]] (Sun386i systems only)

Set or reset breakpoints for debugging. With no arguments, this command displays the existing breakpoints. The *number* argument is a values from 0 to 3, corresponding to the processor debug registers, **DR0** to **DR3**, respectively. Up to 4 distinct breakpoints can be specified. If *number* is not specified then the monitor chooses a breakpoint number. The *breakpoint_virtual_address* argument specifies the breakpoint address. The *type* argument can be one of:

- **x** Instruction Execution breakpoint (the default)
- **m** for Data Write only breakpoint
- **r** Data Reads and Writes only breakpoint.

The *len* argument can be one of: '**b**', '**w**', or '**l**', corresponding to the breakpoint field length of byte, word, or long-word, respectively. The default is '**b**'. Since the breakpoints are set in the on-chip registers, an instruction breakpoint can be placed in ROM code or in code shared by several tasks. If the *number* argument is specified but not *breakpoint_virtual_address*, the corresponding breakpoint is reset.

z [*virtual_address*] (Sun-3 systems only)

Set a breakpoint at *virtual_address* in the address space selected by the **s** command.

FILES

/vmunix

SEE ALSO

eeprom(8S)

NAME

mount, umount – mount and unmount file systems

SYNOPSIS

/usr/etc/mount [**–p**]
/usr/etc/mount –a [**fnv**] [**–t** *type*]
/usr/etc/mount [**–fnrv**] [**–t** *type*] [**–o** *options*] *filesystem directory*
/usr/etc/mount [**–vfn**] [**–o** *options*] *filesystem* | *directory*
/usr/etc/mount –d [**fnvr**] [**–o** *options*] *RFS-resource* | *directory*

/usr/etc/umount [**–t** *type*] [**–h** *host*]
/usr/etc/umount –a [**v**]
/usr/etc/umount [**–v**] *filesystem* | *directory* ...
/usr/etc/umount [**–d**] *RFS-resource* | *directory*

DESCRIPTION

mount attaches a named *filesystem* to the file system hierarchy at the pathname location *directory*, which must already exist. If *directory* has any contents prior to the **mount** operation, these remain hidden until the *filesystem* is once again unmounted. If *filesystem* is of the form *host*:*pathname*, it is assumed to be an NFS file system (type **nfs**).

umount unmounts a currently mounted file system, which can be specified either as a *directory* or a *filesystem*.

mount and **umount** maintain a table of mounted file systems in **/etc/mtab**, described in **fstab**(5). If invoked without an argument, **mount** displays the contents of this table. If invoked with either a *filesystem* or *directory* only, **mount** searches the file **/etc/fstab** for a matching entry, and mounts the file system indicated in that entry on the indicated directory.

mount also allows the creation of new, virtual file systems using **loopback mounts**. Loopback file systems provide access to existing files using alternate pathnames. Once a virtual file system is created, other file systems can be mounted within it without affecting the original file system. File systems that are subsequently mounted onto the original file system, however, are visible to the virtual file system, unless or until the corresponding mount point in the virtual file system is covered by a file system mounted there.

Recursive traversal of loopback mount points is not allowed; after the loopback mount of **/tmp/newroot**, the file **/tmp/newroot/tmp/newroot** does not contain yet another file system hierarchy. Rather, it appears just as **/tmp/newroot** did before the loopback mount was performed (say, as an empty directory).

The standard RC files first perform **4.2** mounts, then **nfs** mounts, during booting. On Sun386i systems, **lo** (loopback) mounts are performed just after **4.2** mounts. **/etc/fstab** files depending on alternate mount orders at boot time will fail to work as expected. Manual modification of **/etc/rc.local** will be needed to make such mount orders work.

See **lofs**(4S) and **fstab**(5) for more information and WARNINGS about loopback mounts.

OPTIONS

mount

–p Print the list of mounted file systems in a format suitable for use in **/etc/fstab**.

–a All. Attempt to mount all the file systems described in **/etc/fstab**. If a *type* argument is specified with **–t**, mount all file systems of that type. Using **–a**, **mount** builds a dependency tree of mount points in **/etc/fstab**. **mount** will correctly mount these file systems regardless of their order in **/etc/fstab** (except loopback mounts; see WARNINGS below).

–f Fake an **/etc/mtab** entry, but do not actually mount any file systems.

–n Mount the file system without making an entry in **/etc/mtab**.

–v Verbose. Display a message indicating each file system being mounted.

-t *type* Specify a file system type. The accepted types are **4.2**, **nfs**, **rfs**, **lo**, **hsfs**, and **tmp**. See **fstab**(5) for a description of **4.2**, **hsfs**, and **nfs**; see **lofs**(4S) for a description of **lo**; and see **tmpfs**(4) for a description of **tmp**. See *System and Network Administration* for details on **rfs**.

-r Mount the specified file system read-only, even if the entry in **/etc/fstab** specifies that it is to be mounted read-write.

Physically write-protected and magnetic-tape file systems must be mounted read-only. Otherwise errors occur when the system attempts to update access times, even if no write operation is attempted.

-d Mount an RFS file system. This option provides compatibility with the System V, Release 3 syntax for RFS mounts. Alternatively, the equivalent Sun syntax, **-t rfs**, may be used.

-o *options*

Specify file system *options*, a comma-separated list of words from the list below. Some options are valid for all file system types, while others apply to a specific type only.

options valid on *all* file systems:

rw|**ro**	Read/write or read-only.
suid|**nosuid**	Setuid execution allowed or disallowed.
grpid	Create files with BSD semantics for the propagation of the group ID. Under this option, files inherit the GID of the directory in which they are created, regardless of the directory's set-GID bit.
noauto	Do not mount this file system that is currently mounted read-only. If the file system is not currently mounted, an error results.
remount	If the file system is currently mounted, and if the entry in **/etc/fstab** specifies that it is to be mounted read-write or **rw** was specified along with **remount**, remount the file system making it read-write. If the entry in **/etc/fstab** specifies that it is to be mounted read-only and **rw** was not specified, the file system is not remounted. If the file system is currently mounted read-write, specifying **ro** along with **remount** results in an error. If the file system is not currently mounted, an error results.

The default is '**rw,suid**'.

options specific to **4.2** file systems:

quota|**noquota**	Usage limits are enforced, or are not enforced. The default is **noquota**.

options specific to **nfs** (NFS) file systems:

bg|**fg**	If the first attempt fails, retry in the background, or, in the foreground.
noquota	Prevent **quota**(1) from checking whether the user is over quota on this file system; if the file system has quotas enabled on the server, quotas will still be checked for operations on this file system.
retry=*n*	The number of times to retry the mount operation.
rsize=*n*	Set the read buffer size to *n* bytes.
wsize=*n*	Set the write buffer size to *n* bytes.
timeo=*n*	Set the NFS timeout to *n* tenths of a second.
retrans=*n*	The number of NFS retransmissions.
port=*n*	The server IP port number.
soft|**hard**	Return an error if the server does not respond, or continue the retry request until the server responds.
intr	Allow keyboard interrupts on hard mounts.
secure	Use a more secure protocol for NFS transactions.
posix	Request POSIX.1 semantics for the file system. Requires a mount version 2 **mountd**(8C) on the server.

acregmin=*n* Hold cached attributes for at least *n* seconds after file modification.
acregmax=*n* Hold cached attributes for no more than *n* seconds after file modification.
acdirmin=*n* Hold cached attributes for at least *n* seconds after directory update.
acdirmax=*n* Hold cached attributes for no more than *n* seconds after directory update.
actimeo=*n* Set *min* and *max* times for regular files and directories to *n* seconds.
nocto Suppress fresh attributes when opening a file.
noac Suppress attribute and name (lookup) caching.

Regular defaults are:

fg,retry=10000,timeo=7,retrans=3,port=NFS_PORT,hard,
acregmin=3,acregmax=60,acdirmin=30,acdirmax=60

actimeo has no default; it sets **acregmin**, **acregmax**, **acdirmin** and **acdirmax**

Defaults for **rsize** and **wsize** are set internally by the system kernel.

options specific to **rfs** (RFS) file systems:

bg | fg If the first attempt fails, retry in the background, or, in the foreground.
retry=*n* The number of times to retry the mount operation.

Defaults are the same as for NFS.

umount

–h *host* Unmount all file systems listed in **/etc/mtab** that are remote-mounted from *host*.

–t *type* Unmount all file systems listed in **/etc/mtab** that are of a given *type*.

–a Unmount all file systems currently mounted (as listed in **/etc/mtab**).

–v Verbose. Display a message indicating each file system being unmounted.

–d Unmount an RFS file system. This option provides compatibility with the System V, Release 3 syntax for unmounting an RFS file system.

NFS FILESYSTEMS

Background vs. Foreground

Filesystems mounted with the **bg** option indicate that **mount** is to retry in the background if the server's mount daemon (mountd(8C)) does not respond. **mount** retries the request up to the count specified in the **retry=*n*** option. Once the file system is mounted, each NFS request made in the kernel waits **timeo=*n*** tenths of a second for a response. If no response arrives, the time-out is multiplied by 2 and the request is retransmitted. When the number of retransmissions has reached the number specified in the **retrans=*n*** option, a file system mounted with the **soft** option returns an error on the request; one mounted with the **hard** option prints a warning message and continues to retry the request.

Read-Write vs. Read-Only

File systems that are mounted **rw** (read-write) should use the **hard** option.

Interrupting Processes With Pending NFS Requests

The **intr** option allows keyboard interrupts to kill a process that is hung while waiting for a response on a hard-mounted file system.

Quotas

Quota checking on NFS file systems is performed by the server, not the client; if the file system has the **quota** option on the server, quota checking is performed for both local requests and NFS requests. When a user logs in, **login**(1) runs the **quota**(1) program to check whether the user is over their quota on any of the file systems mounted on the machine. This check is performed for NFS file systems by an RPC call to the **rquotad**(8C) server on the machine from which the file system is mounted. This can be time-consuming, especially if the remote machine is down. If the **noquota** option is specified for an NFS file system, **quota** will not check whether the user is over their quota on that file system, which can speed up the process of logging in. This does *not* disable quota checking for operations on that file system; it merely disables reporting whether the user is over quota on that file system.

Secure Filesystems

The **secure** option must be given if the server requires secure mounting for the file system.

File Attributes

The attribute cache retains file attributes on the client. Attributes for a file are assigned a time to be flushed. If the file is modified before the flush time, then the flush time is extended by the time since the last modification (under the assumption that files that changed recently are likely to change soon). There is a minimum and maximum flush time extension for regular files and for directories. Setting **actimeo=***n* extends flush time by *n* seconds for both regular files and directories.

SYSTEM V COMPATIBILITY

System V File-Creation Semantics

Ordinarily, when a file is created its GID is set to the effective GID of the calling process. This behavior may be overridden on a per-directory basis, by setting the set-GID bit of the parent directory; in this case, the GID is set to the GID of the parent directory (see **open**(2V) and **mkdir**(2V)). Files created on file systems that are mounted with the **grpid** option will obey BSD semantics; that is, the GID is unconditionally inherited from that of the parent directory.

EXAMPLES

To mount a local disk:

mount /dev/xy0g /usr

To fake an entry for **nd** root:

mount –ft 4.2 /dev/nd0 /

To mount all 4.2 file systems:

mount –at 4.2

To mount a remote file system:

mount –t nfs serv:/usr/src /usr/src

To mount a remote file system:

mount serv:/usr/src /usr/src

To hard mount a remote file system:

mount –o hard serv:/usr/src /usr/src

To mount an RFS remote file system, retrying in the background on failure:

mount –d –o bg SRC /usr/src

To mount an RFS remote file system read-only:

mount –t rfs –r SRC /usr/src

To save current mount state:

mount –p > /etc/fstab

Note: this is not recommended when running the automounter, see **automount**(8).

To loopback mount file systems:

mount –t lo /export/tmp/localhost /tmp
mount –t lo /export/var/localhost /var lo
mount –t lo /export/cluster/sun386.sunos4.0.1 /usr/cluster
mount –t lo /export/local/sun386 /usr/local

FILES

/etc/mtab	table of mounted file systems
/etc/fstab	table of file systems mounted at boot

WARNINGS

mount does not understand the mount order dependencies involved in loopback mounting. Loopback mounts may be dependent on two mounts having been previously performed, while **nfs** and **4.2** mounts are dependent only on a single previous mount. As a rule of thumb, place loopback mounts at the end of the **/etc/fstab** file. See **lofs**(4S) for a complete description.

SEE ALSO

mkdir(2V), **mount**(2V), **open**(2V), **unmount**(2V), **lofs**(4S), **fstab**(5), **mtab**(5), **automount**(8), **mountd**(8C), **nfsd**(8)

BUGS

Mounting file systems full of garbage crashes the system.

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 being mounted on top of the symbolic link itself.

NAME

mountd, rpc.mountd – NFS mount request server

SYNOPSIS

/usr/etc/rpc.mountd [–n]

AVAILABILITY

This program is available with the *Networking* software installation option. Refer to *Installing SunOS 4.1* for information on how to install optional software.

DESCRIPTION

mountd is an RPC server that answers file system mount requests. It reads the file **/etc/xtab**, described in **exports**(5), to determine which file systems are available for mounting by which machines. It also provides information as to what file systems are mounted by which clients. This information can be printed using the **showmount**(8) command.

The **mountd** daemon is normally invoked by **rc**(8).

OPTIONS

–n Do not check that the clients are root users. Though this option makes things slightly less secure, it does allow older versions (pre-3.0) of client NFS to work.

FILES

/etc/xtab

SEE ALSO

exports(5), **rc**(8), **showmount**(8)

NAME

mount_tfs, umount_tfs – mount and dismount TFS filesystems

SYNOPSIS

/usr/etc/mount_tfs [**–r**] *fs1 fs2 ... fsN dir*
/usr/etc/mount –t tfs [**–o** *options*] *fs dir*

/usr/etc/umount_tfs *dir*
/usr/etc/umount *dir*

DESCRIPTION

mount_tfs attaches a translucent file service (TFS) filesystem to the directory *dir*. After the mount, the directory *dir* is a TFS directory whose frontmost directory is *fs1* and whose backmost directory is *dir*, with any number of directories intervening. Effectively, the directories *fs1 ... fsN* are stacked in front of *dir*.)

TFS filesystems can also be mounted using the **mount**(8) command. The **mount** command can only mount one directory, *fs*, in front of the backmost directory, *dir*.

umount_tfs detaches the TFS filesystem rooted at *dir*. See **tfs**(4S) for a description of a TFS filesystem.

OPTIONS

–r Mount the TFS filesystem read-only.

SEE ALSO

lsw(1), **unwhiteout**(1), **tfs**(4S), **mount**(8), **tfsd**(8)

BUGS

mount_tfs will cause **tfsd**(8) to deadlock (hang and answer no more requests) if it is used in conjunction with Network Software Environment (NSE) execsets. For example, a deadlock will occur if a user has used **mount_tfs** to mount over **/usr/lib**, and then tries to activate an NSE environment whose execset mounts over **/usr/lib**.

The directories *fs1*, *fs2*, ..., *fsN* must be writable.

NAME

named, in.named – Internet domain name server

SYNOPSIS

/usr/etc/in.named [**–d** *level*] [**–p** *port*] [[**–b**] *bootfile*]

DESCRIPTION

named is the Internet domain name server. It is used by resolver libraries to provide access to the Internet distributed naming database. The domain name server is described in the *System and Network Administration*. See RFC 1034 and RFC 1035 for more details. With no arguments **named** reads **/etc/named.boot** for any initial data, and listens for queries on a privileged port.

OPTIONS

–d *level* Print debugging information. *level* is a number indicating the level of messages printed.

–p *port* Use *port* as the port number, rather than the standard port number.

–b *bootfile*
Use *bootfile* rather than **/etc/named.boot**.

EXAMPLE

```
;
;          boot file for name server
;
; type              domain              source file or host
;
primary             berkeley.edu    named.db
secondary           cc.berkeley.edu 10.2.0.78 128.32.0.10
cache               .               named.ca
```

The **primary** line states that the file **named.db** contains authoritative data for **berkeley.edu**. The file **named.db** contains data in the master file format, described in RFC 1035, except that all domain names are relative to the origin; in this case, **berkeley.edu** (see below for a more detailed description).

The **secondary** line specifies that all authoritative data under **cc.berkeley.edu** is to be transferred from the name server at **10.2.0.78**. If the transfer fails it will try **128.32.0.10**, and continue for up to 10 tries at that address. The secondary copy is also authoritative for the domain.

The **cache** line specifies that data in **named.ca** is to be placed in the cache (only used to find the root domain servers). The file **named.ca** is in the same format as **named.db**.

The master file consists of entries of the form:

$INCLUDE *<filename>*
$ORIGIN *<domain>*
<domain> <opt_ttl> <opt_class> <type> <resource_record_data>

where *domain* is '.' for the root, '@' for the current origin, or a standard domain name. If *domain* is a standard domain name that does not end with '.', the current origin is appended to the domain. Domain names ending with '.' are unmodified.

The *opt_ttl* field is an optional integer number for the time-to-live field. It defaults to zero.

The *opt_class* field is currently one token, 'IN' for the Internet.

The *type* field is one of the following tokens; the data expected in the *resource_record_data* field is in parentheses.

A A host address (dotted quad).

NS An authoritative name server (domain).

MX A mail exchanger (domain).

CNAME
The canonical name for an alias (domain).

SOA Marks the start of a zone of authority (5 numbers). (see RFC 1035)).

MB A mailbox domain name (domain).

MG A mail group member (domain).

MR A mail rename domain name (domain).

NULL A null resource record (no format or data).

WKS A well know service description (not implemented yet).

PTR A domain name pointer (domain).

HINFO Host information (cpu_type OS_type).

MINFO Mailbox or mail list information (request_domain error_domain).

FILES

/etc/named.boot	name server configuration boot file
/etc/named.pid	the process ID
/var/tmp/named.run	debug output
/var/tmp/named_dump.db	dump of the name servers database

SEE ALSO

kill(1), **signal**(3V), **resolver**(3), **resolv.conf**(5), **nslookup**(8C)

System and Network Administration

Mockapetris, Paul, *Domain Names - Concepts and Facilities*, RFC 1034, Network Information Center, SRI International, Menlo Park, Calif., November 1987.

Mockapetris, Paul, *Domain Names - Implementation and Specification*, RFC 1035, Network Information Center, SRI International, Menlo Park, Calif., November 1987.

Mockapetris, Paul, *Domain System Changes and Observations*, RFC 973, Network Information Center, SRI International, Menlo Park, Calif., January 1986.

Partridge, Craig, *Mail Routing and the Domain System*, RFC 974, Network Information Center, SRI International, Menlo Park, Calif., January 1986.

NOTES

The following signals have the specified effect when sent to the server process using the **kill**(1) command.

SIGHUP Causes server to read named.boot and reload database.

SIGINT Dumps current data base and cache to **/var/tmp/named_dump.db**.

SIGUSR1

Turns on debugging; each subsequent **SIGUSR1** increments debug level.

SIGUSR2

Turns off debugging completely.

NAME

ncheck – generate names from i-numbers

SYNOPSIS

/usr/etc/ncheck [**–i** *numbers*] [**–as**] *filesystem*

DESCRIPTION

Note: For most normal file system maintenance, the function of **ncheck** is subsumed by **fsck**(8).

ncheck generates a pathname versus i-number list of files for the indicated *filesystem*. Names of directory files are followed by '.'

The report is in no useful order, and probably should be sorted.

OPTIONS

–i *numbers*
Report only those files whose i-*numbers* follow.

–a Print the names '.' and '..', which are ordinarily suppressed.

–s Report only special files and files with set-user-ID mode. This is intended to discover concealed violations of security policy.

SEE ALSO

sort(1V), **dcheck**(8), **fsck**(8), **icheck**(8)

DIAGNOSTICS

When the filesystem structure is improper, '**??**' denotes the "parent" of a parentless file and a pathname beginning with '**...**' denotes a loop.

NAME

ndbootd – ND boot block server

SYNOPSIS

ndbootd [**–dv**]

DESCRIPTION

ndbootd sends boot blocks to diskless Sun-2 system clients that request them using the (now obsolete) ND protocol. This server uses the boot block contained in the file **/tftpboot/sun2.bb**. A client must appear in the **ethers**(5) and **hosts**(5) databases, in order for the request to be served. In determining whether to serve the client, **ndbootd** checks the **/tftpboot** directory for a file whose name is the client's IP address in hexadecimal notation. For example, if the file **/tftpboot/C00901AD** exists, the machine at IP address 192.9.1.173 can be served. This file normally contains the boot program that is sent to the client by **tftpd**(8C).

Only root can invoke **ndbootd**.

OPTIONS

–d Debug. Display information about ignored packets, retransmissions, and address translation.

–v Verbose. Show a detailed listing of packets sent and received, etc.

If either option is used, all output is sent to the invoking terminal. Otherwise, error output (if any) appears on the console.

FILES

/tftpboot	bootfiles directory
/tftpboot/sun2.bb	boot blocks
/tftpboot/????????	boot programs for clients

SEE ALSO

ethers(5), **hosts**(5), **boot**(8S), **tftpd**(8C)

NAME

netconfig – PNP boot service

SYNOPSIS

/single/netconfig [**–e**] [**–n**]

AVAILABILITY

Available only on Sun 386i systems running a SunOS 4.0.*x* release or earlier. Not a SunOS 4.1 release feature.

DESCRIPTION

netconfig is used both for automatic installation of new diskful systems, and during routine booting of all systems. The sequence of actions taken by **netconfig** depends on which of these situations is in effect, but it always sets the hostname, domainname, time, timezone, and interface IP address. If the system is newly installed on the network, it does more, perhaps interrogating the user about system configuration.

netconfig is invoked with the **–e** option from the **/etc/rc.boot** script.

Invoked without options, **netconfig** may perform PNP set up, including set up of files, passwords, and secure RPCs. Unless **–n** is specified, it writes **/etc/net.conf**, which is read later by **rc.boot**. This includes the VERBOSE flag, derived from NVRAM data, which controls the verbosity of the commands in **rc.boot**.

Routine Booting

Boot servers use information stored locally in Network Interface Service (NIS) acquiring it over the network, except that they get the time from the *timehost* system if it is up. The following describes the steps taken by boot clients: diskful clients, diskless clients, and network clients.

Boot clients first invoke **rarp** to acquire an IP address. This is followed by a ICMP **Netmask** request to obtain the IP subnetwork mask, and then a PNP_WHOAMI RPC to determine the system's name, NIS domain, and time zone. Then the systems clock is set using the RFC 868 time service. If PNP_WHOAMI fails, a PNP_SETUP sequence is followed by set up of **/etc/passwd** and other files.

OPTIONS

–e Check shell environment variables. This option is specified during routine boot. HOSTNAME and DOMAINNAME are used to determine if the system is an NIS server using local NIS maps. Otherwise, if NETWORKED is YES, **netconfig** probes the network for network configuration. MUST_SETUP requires writing **/etc/passwd** and other files for setup in restricted network environments.

–n Used in conjunction with '**–e**', this does not probe the network for anything but just sets the hostname and domainname of the system from the environment variables HOSTNAME and DOMAINNAME respectively. Does not write the **/etc/net.conf** file.

FILES

/var/yp/*domainname***/netmasks**
/var/yp/*domainname***/hosts**

SEE ALSO

pnp(3R), **pnpboot**(8C), **pnpd**(8C), **rarpd**(8C)

NOTES

The Network Information Service (NIS) was formerly known as Sun Yellow Pages (YP). The functionality of the two remains the same; only the name has changed.

NAME

netstat – show network status

SYNOPSIS

netstat [**–aAn**] [**–f** *address_family*] [**system**] [**core**]

netstat [**–n**] [**–s**] [**–m** | **–i** | **–r**] [**–f** *address_family*] [**system**] [**core**]

netstat [**–n**] [**–I** *interface*] *interval* [**system**] [**core**]

AVAILABILITY

This program is available with the *Networking* software installation option. Refer to *Installing SunOS 4.1* for information on how to install optional software.

DESCRIPTION

netstat displays the contents of various network-related data structures in various formats, depending on the options you select.

The first form of the command displays a list of active sockets for each protocol. The second form selects one from among various other network data structures. The third form displays running statistics of packet traffic on configured network interfaces; the *interval* argument indicates the number of seconds in which to gather statistics between displays.

The default value for the **system** argument is **/vmunix**; for *core*, the default is **/dev/kmem**.

OPTIONS

–a Show the state of all sockets; normally sockets used by server processes are not shown.

–A Show the address of any protocol control blocks associated with sockets; used for debugging.

–f *address_family*
Limit statistics or address control block reports to those of the specified *address_family*, which can be one of:

inet	For the AF_INET address family, or
unix	For the AF_UNIX family.

–i Show the state of interfaces that have been auto-configured. Interfaces that are statically configured into a system, but not located at boot time, are not shown.

–I *interface* Highlight information about the indicated *interface* in a separate column; the default (for the third form of the command) is the interface with the most traffic since the system was last rebooted. *interface* can be any valid interface listed in the system configuration file, such as **ie0** or **le0**.

–m Show the statistics recorded by management routines for the network's private buffer pool.

–n Show network addresses as numbers. **netstat** normally displays addresses as symbols. This option may be used with any of the display formats.

–r Show the routing tables. (When **–s** is also present, show routing statistics instead.)

–s Show per-protocol statistics. When used with the **–r** option, show routing statistics.

–t Replace queue length information with timer information.

DISPLAYS

Active Sockets (First Form)

The display for each active socket shows the local and remote address, the send and receive queue sizes (in bytes), the protocol, and the internal state of the protocol.

The symbolic format normally used to display socket addresses is either:

hostname.port

when the name of the host is specified, or:

network.port

if a socket address specifies a network but no specific host. Each *hostname* and *network* is shown according to its entry in the **/etc/hosts** or the **/etc/networks** file, as appropriate.

If the network or hostname for an address is not known (or if the **–n** option is specified), the numerical network address is shown. Unspecified, or ''wildcard'', addresses and ports appear as ''*''. (For more information regarding the Internet naming conventions, refer to **inet**(3N)).

TCP Sockets

The possible state values for TCP sockets are as follows:

CLOSED	Closed: the socket is not being used.
LISTEN	Listening for incoming connections.
SYN_SENT	Actively trying to establish connection.
SYN_RECEIVED	Initial synchronization of the connection under way.
ESTABLISHED	Connection has been established.
CLOSE_WAIT	Remote shut down: waiting for the socket to close.
FIN_WAIT_1	Socket closed, shutting down connection.
CLOSING	Closed, then remote shutdown: awaiting acknowledgement.
LAST_ACK	Remote shut down, then closed: awaiting acknowledgement.
FIN_WAIT_2	Socket closed, waiting for shutdown from remote.
TIME_WAIT	Wait after close for remote shutdown retransmission.

Network Data Structures (Second Form)

The form of the display depends upon which of the **–m**, **–i**, **–h** or **–r**, options you select. (If you specify more than one of these options, **netstat** selects one in the order listed here.)

Routing Table Display

The routing table display lists the available routes and the status of each. Each route consists of a destination host or network, and a gateway to use in forwarding packets. The *flags* column shows the status of the route (**U** if ''up''), whether the route is to a gateway (**G**), and whether the route was created dynamically by a redirect (**D**).

Direct routes are created for each interface attached to the local host; the gateway field for such entries shows the address of the outgoing interface.

The **refcnt** column gives the current number of active uses per route. (Connection-oriented protocols normally hold on to a single route for the duration of a connection, whereas connectionless protocols obtain a route while sending to the same destination.)

The **use** column displays the number of packets sent per route.

The *interface* entry indicates the network interface utilized for the route.

Cumulative Traffic Statistics (Third Form)

When the *interval* argument is given, **netstat** displays a table of cumulative statistics regarding packets transferred, errors and collisions, the network addresses for the interface, and the maximum transmission unit (''mtu''). The first line of data displayed, and every 24th line thereafter, contains cumulative statistics from the time the system was last rebooted. Each subsequent line shows incremental statistics for the *interval* (specified on the command line) since the previous display.

SEE ALSO

hosts(5), **networks**(5), **protocols**(5), **services**(5) **iostat**(8), **trpt**(8C), **vmstat**(8)

BUGS

The notion of errors is ill-defined. Collisions mean something else for the IMP.

The kernel's tables can change while **netstat** is examining them, creating incorrect or partial displays.

NAME

newaliases – rebuild the data base for the mail aliases file

SYNOPSIS

newaliases

DESCRIPTION

newaliases rebuilds the random access data base for the mail aliases file **/etc/aliases**. It is run automatically by **sendmail**(8) (in the default configuration) whenever a message is sent.

FILES

/etc/aliases

SEE ALSO

aliases(5), **sendmail**(8)

NAME

newfs – create a new file system

SYNOPSIS

/usr/etc/newfs [–Nv] [*mkfs-options*] *raw-special-device*

DESCRIPTION

newfs is a "friendly" front-end to the **mkfs**(8) program. On Sun systems, the disk type is determined by reading the disk label for the specified *raw-special-device*.

raw-special-device is the name of a raw special device residing in **/dev**, including the disk partition, where you want the new file system to be created. If you want to make a file system on **sd0[a-h]**, specify **sd0[a-h]**, **rsd0[a-h]** or **/dev/rsd0[a-h]**; if you only specify **sd0[a-h]**, **newfs** will find the proper device.

newfs then calculates the appropriate parameters to use in calling **mkfs**, and builds the file system by forking **mkfs**.

You must be super-user to use this command.

OPTIONS

–N Print out the file system parameters without actually creating the file system.

–v Verbose. **newfs** prints out its actions, including the parameters passed to **mkfs**.

mkfs-options

Options that override the default parameters passed to **mkfs**(8) are:

–a *apc* Number of alternates per cylinder (SCSI devices only).

–b *block-size*

The block size of the file system in bytes. The default is 8192.

–c *#cylinders/group*

The number of cylinders per cylinder group in a file system. The default is 16.

–d *rotdelay*

This specifies 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.

–f *frag-size*

The fragment size of the file system in bytes. The default is 1024.

–i *bytes/inode*

This specifies the density of inodes in the file system. The default is 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.

–m *free-space%*

The percentage of space reserved from normal users; the minimum free space threshold. The default is 10%.

–o *optimization*

(**space** or **time**). The file system can either be instructed to try to minimize the time spent allocating blocks, or to try to minimize the space fragmentation on the disk. If the minimum free space threshold (as specified by the **–m** option) is less than 10%, the default is to optimize for **space**; if the minimum free space threshold is greater than or equal to 10%, the default is to optimize for **time**.

–r *revolutions/minute*

The speed of the disk in revolutions per minute (normally 3600).

–s *size* The size of the file system in sectors.

–t *#tracks/cylinder*
The number of tracks per cylinders on the disk. The default is 16.

–n *#rotational-positions*
The number of distinguished rotational positions. The default is 8.

EXAMPLES

The following example verbosely displays the parameters for the raw special device, **sd0a**, but does not actually create a new file system:

```
example% /usr/etc/newfs –vN sd0a
mkfs –N /dev/rsd0a 16048 34 8 8192 1024 16 10 60 2048 t 0 -1
/dev/rsd0a:     16048 sectors in 59 cylinders of 8 tracks, 34 sectors
        8.2Mb in 4 cyl groups (16 c/g, 2.23Mb/g, 896 i/g)
super-block backups (for fsck –b#) at:
 32, 4432, 8832, 13232,
example%
```

SEE ALSO

fs(5), **fsck**(8), **installboot**(8S), **mkfs**(8), **tunefs**(8)

System and Network Administration

DIAGNOSTICS

newfs: *special* **No such file or directory**
The device specified does not exist, or a disk partition was not specified.

special: **cannot open**
You must be super-user to use this command.

NOTES

To install the bootstrap programs for a root partition, run **installboot**(8S) after **newfs**.

NAME

newkey – create a new key in the publickey database

SYNOPSIS

newkey –**h** *hostname*
newkey –**u** *username*

DESCRIPTION

newkey is normally run by the network administrator on the Network Interface Service (NIS) master machine in order to establish public keys for users and super-users on the network. These keys are needed for using secure RPC or secure NFS.

newkey will prompt for the login password of the given username and then create a new public/secret key pair in **/etc/publickey** encrypted with the login password of the given user.

Use of this program is not required: users may create their own keys using **chkey**(1).

OPTIONS

–**h** *hostname* Create a new public key for the super-user at the given hostname. Prompts for the root password of the given hostname.

–**u** *username* Create a new public key for the given username. Prompts for the NIS password of the given username.

SEE ALSO

chkey(1), **keylogin**(1), **publickey**(5), **keyserv**(8C)

NOTES

The Network Information Service (NIS) was formerly known as Sun Yellow Pages (YP). The functionality of the two remains the same; only the name has changed.

NAME

nfsd, biod – NFS daemons

SYNOPSIS

/usr/etc/nfsd [*nservers*]

/usr/etc/biod [*nservers*]

DESCRIPTION

nfsd starts the daemons that handle client filesystem requests. *nservers* is the number of file system request daemons to start. This number should be based on the load expected on this server. Eight seems to be a good number.

biod starts *nservers* asynchronous block I/O daemons. This command is used on a NFS client to buffer cache handle read-ahead and write-behind. The magic number for *nservers* in here is also eight.

When a file that is opened by a client is unlinked (by the server), a file with a name of the form **.nfs***XXX* (where *XXX* is a number) is created by the client. When the open file is closed, the **.nfs***XXX* file is removed. If the client crashes before the file can be closed, the **.nfs***XXX* file is not removed.

FILES

.nfs*XXX*	client machine pointer to an open-but-unlinked file

SEE ALSO

exports(5), **mountd**(8C)

NAME

nfsstat – Network File System statistics

SYNOPSIS

nfsstat [–cmnrsz]

AVAILABILITY

This program is available with the *Networking* software installation option. Refer to *Installing SunOS 4.1* for information on how to install optional software.

DESCRIPTION

nfsstat displays statistical information about the NFS (Network File System) and RPC (Remote Procedure Call), interfaces to the kernel. It can also be used to reinitialize this information. If no options are given the default is

nfsstat –cnrs

That is, display everything, but reinitialize nothing.

OPTIONS

–c Display client information. Only the client side NFS and RPC information will be printed. Can be combined with the **–n** and **–r** options to print client NFS or client RPC information only.

–m Display statistics for each NFS mounted file system. This includes the server name and address, mount flags, current read and write sizes, the retransmission count, and the timers used for dynamic retransmission.

–n Display NFS information. NFS information for both the client and server side will be printed. Can be combined with the **–c** and **–s** options to print client or server NFS information only.

–r Display RPC information.

–s Display server information.

–z Zero (reinitialize) statistics. This option is for use by the super-user only, and can be combined with any of the above options to zero particular sets of statistics after printing them.

DISPLAYS

The server RPC display includes the fields:

calls total number of RPC calls received

badcalls total number of calls rejected

nullrecv number of times no RPC packet was available when trying to receive

badlen number of packets that were too short

xdrcall number of packets that had a malformed header

The server NFS display shows the number of NFS calls received (**calls**) and rejected (**badcalls**), and the counts and percentages for the various calls that were made.

The client RPC display includes the following fields:

calls total number of RPC calls sent
badcalls total of calls rejected by a server
retrans number of times a call had to be retransmitted
badxid number of times a reply did not match the call
timeout number of times a call timed out
wait number of times a call had to wait on a busy CLIENT handle
newcred number of times authentication information had to be refreshed

The client NFS display shows the number of calls sent and rejected, as well as the number of times a CLIENT handle was received (**nclget**), the number of times a call had to sleep while awaiting a handle (**nclsleep**), as well as a count of the various calls and their respective percentages.

FILES

/vmunix	system namelist
/dev/kmem	kernel memory

NAME

listen, nlsadmin – network listener service administration for RFS

SYNOPSIS

nlsadmin [**–mx**] [**–edr** *service_code net_spec*] [**–ikqsv** *net_spec*]
[**–lt** *addr net_spec*] [**–a** *service_code* [**–p** *modules*] **–c** *command* **–y** *comment net_spec*]
[**–qz** *code net_spec*] [**–z** *code net_spec*] [*net_spec*]

/usr/etc/listen

AVAILABILITY

This program is available with the *RFS* software installation option. Refer to *Installing SunOS 4.1* for information on how to install optional software.

DESCRIPTION

nlsadmin configures, initiates and terminates network listener (**listen**) servers for the local host. Each network (transport provider) has an associated **listen** daemon to service it locally. The **listen** daemon for each is configured separately. A **listen** daemon accepts network service requests when they arrive, and spawns servers in response to those requests. It can be used on any network (transport provider) that conforms to the transport provider specification.

nlsadmin can also report on the listener processes on a machine, either individually (per network) or collectively.

Changing the list of services provided by the listener produces immediate changes, while changing an address on which the listener listens has no effect until the listener is restarted.

nlsadmin without any options gives a brief usage message.

The *net_spec* argument to **nlsadmin** refers to a particular **listen** daemon. Specifically, *net_spec* is the relative path name of the entry under **/dev** for a given network.

–x Report the status of all of the listener processes installed on this machine.

–e *service_code net_spec*
–d *service_code net_spec*
Enable or disable, respectively, the service indicated by *service_code* for the specified network. The service must have previously been added to the listener for that network (see the **–a** option). When a listener is disabled, processes serving prior requests continue until they complete.

–r *service_code net_spec*
Remove the entry for the *service_code* from that listener's list of services.

–i *net_spec* Initialize or change a listener process for the network specified by *net_spec*. That is, create and initialize the files required by the listener. Initializing a listener with this option does not start it running. The listener must be initialized before assigning addressing or services. Note: the listener should only be initialized once for a given network.

–q *net_spec* Query the status of the listener process for the specified network. If the listener process is active, **nlsadmin** exits with a status of 0. If no such process is active, the exit code is 1. The exit code will be greater than 1 if there is an error.

–s *net_spec*
–k *net_spec*
Start or kill, respectively, the listener process for the indicated network. When a listener is killed, processes that are still running as a result of prior service requests will continue unaffected. The listener runs under its own ID of **listen** with group ID (GID) **adm**. This GID appear in the system password file **/etc/passwd**; the **HOME** directory listed for the GID is concatenated with *net_spec* to determine the location of the listener configuration information for each network.

nlsadmin may be invoked by any user to generate reports, but all operations that affect a listener's status or configuration are restricted to the super-user.

−v *net_spec* Verbose. Report on the servers associated with *net_spec*, giving the service code, status, command, and comment for each.

−l *addr net_spec* Change or set the address for the general listener service. This is the address generally used by remote processes to access the servers available through the listener (see the **−a** option). *addr* is the transport address on which to listen, and is interpreted using a syntax that allows for a variety of address formats. By default *addr* is interpreted as the symbolic ASCII representation of the transport address. An *addr* preceded by a '\x' (BACKSLASH-X) lets you enter an address in hexadecimal notation. Note: *addr* must be quoted if it contains any blanks. If *addr* is just a dash ('—'), **nlsadmin** merely reports the currently configured address.

A change of address does not take effect until the next time the listener for that network is started.

−t *addr net_spec* Change or set the address on which the listener listens for requests for terminal service. Otherwise, this is similar to **−l**. A terminal service address should not be defined unless the appropriate remote login software is available; if such software is available, it must be configured as service code **1** (see the **−a** option).

[**−m**] **−a** *service_code* **−c** *cmd* **−y** *comment net_spec*
Add a new service to the list of services available through the indicated listener. *service_code* is the code for the service, *cmd* is the command to be invoked in response to that service code, comprised of the full path name of the server and its arguments, and *comment* is a brief (free-form) description of the service for use in various reports. Note: *cmd* must be quoted if it contains arguments for the server. Similarly, *comment* must also be quoted, so as to appear to be a single word to the shell. When a service is added, it is initially enabled (see the **−e** and **−d** options).

If the **−m** option is specified, the entry is marked as an administrative entry. Service codes **1** through **100** are reserved for administrative entries, which are those that require special handling internally. In particular, code **1** is assigned to the remote login service, which is the service automatically invoked for connections to the terminal login address.

A service must explicitly be added to the listener for each network on which that service is to be available. This operation is normally performed only when the service is installed on a machine, or when populating the list of services for a new network.

−qz *code net_spec*
Query the status of the service with service code *code* on network *net_spec*, Exit with a status of 0 if the service is enabled, 1 if the service is disabled, or greater than 1 on error.

−z *code net_spec* Print a report on the server associated with *net_spec* that has service code *code*, giving the same information as in the **−v** option.

net_spec Print the status of the listener process for *net_spec*.

DIAGNOSTICS

If the command is not run under the proper ID, an error message is sent to the standard error, and the command terminates.

FILES

/usr/etc/listen
/usr/net/nls/*net_spec*

SEE ALSO

Network Programming

NAME

nslookup – query domain name servers interactively

SYNOPSIS

nslookup [**–l**] [*address*]

DESCRIPTION

nslookup is an interactive program to query Internet domain name servers. The user can contact servers to request information about a specific host or print a list of hosts in the domain.

OPTIONS

–l Use the local host's name server instead of the servers in **/etc/resolv.conf**. (If **/etc/resolv.conf** does not exist or does not contain server information, the **–l** option does not have any effect).

address Use the name server on the host machine with the given Internet address.

USAGE

Overview

The Internet domain name-space is tree-structured, with top-level domains such as:

COM	commercial establishments
EDU	educational institutions
GOV	government agencies
MIL	MILNET hosts

If you are looking for a specific host, you need to know something about the host's organization in order to determine the top-level domain it belongs to. For instance, if you want to find the Internet address of a machine at UCLA, do the following:

- Connect with the root server using the **root** command. The root server of the name space has knowledge of the top-level domains.
- Since UCLA is a university, its domain name is **ucla.edu**. Connect with a server for the **ucla.edu** domain with the command **server**ucla.edu. The response will print the names of hosts that act as servers for that domain. Note: the root server does not have information about **ucla.edu**, but knows the names and addresses of hosts that do. Once located by the root server, all future queries will be sent to the UCLA name server.
- To request information about a particular host in the domain (for instance, **locus**), just type the host name. To request a listing of hosts in the UCLA domain, use the **ls** command. The **ls** command requires a domain name (in this case, **ucla.edu**) as an argument.

Note: if you are connected with a name server that handles more than one domain, all lookups for host names must be fully specified with its domain. For instance, the domain **harvard.edu** is served by **seismo.css.gov**, which also services the **css.gov** and **cornell.edu** domains. A lookup request for the host **aiken** in the **harvard.edu** domain must be specified as **aiken.harvard.edu.** However, the

set *domain*=**name**

and

set defname

commands can be used to automatically append a domain name to each request.

After a successful lookup of a host, use the **finger** command to see who is on the system, or to finger a specific person. To get other information about the host, use the

set *querytype*=**value**

command to change the type of information desired and request another lookup. (**finger** requires the type to be A.)

Commands

Commands may be interrupted at any time by typing CTRL-C. To exit, type CTRL-D (EOF). The command line length must be less than 80 characters. Note: an unrecognized command will be interpreted as a host name.

host [*server*]
Look up information for *host* using the current default server or using *server* if it is specified.

server *domain*
lserver *domain*
Change the default server to *domain*. **lserver** uses the initial server to look up information about *domain* while **server** uses the current default server. If an authoritative answer can't be found, the names of servers that might have the answer are returned.

root Changes the default server to the server for the root of the domain name space. Currently, the host **sri-nic.arpa** is used; this command is a synonym for '**lserver sri-nic.arpa**'.) The name of the root server can be changed with the **set root** command.

finger [*name*]
Connect with the finger server on the current host, which is defined by a previous successful lookup for a host's address information (see the **set** *querytype*=A command). As with the shell, output can be redirected to a named file using > and >>.

"ls [–ah]
List the information available for *domain*. The default output contains host names and their Internet addresses. The **–a** option lists aliases of hosts in the domain. The **–h** option lists CPU and operating system information for the domain. As with the shell, output can be redirected to a named file using > and >>. When output is directed to a file, hash marks are printed for every 50 records received from the server.

view*filename*
Sort and list the output of the **ls** command with **more**(1).

help

? Print a brief summary of commands.

set*keyword* [= *value*] This command is used to change state information that affects the lookups. Valid keywords are:

all Prints the current values of the various options to **set**. Information about the current default server and host is also printed.

[no]deb[ug]
Turn debugging mode on. A lot more information is printed about the packet sent to the server and the resulting answer. The default is **nodebug**.

[no]def[name]
Append the default domain name to every lookup. The default is **nodefname.**

do[main]=*filename*
Change the default domain name to *filename*. The default domain name is appended to all lookup requests if **defname** option has been set. The default is the value in **/etc/resolv.conf**.

q[querytype]=*value*
Change the type of information returned from a query to one of:

A The host's Internet address (the default).
CNAME
The canonical name for an alias.
HINFO The host CPU and operating system type.
MD The mail destination.

MX The mail exchanger.
MB The mailbox domain name.
MG The mail group member.
MINFO The mailbox or mail list information.

(Other types specified in the RFC883 document are valid, but are not very useful.)

[no]recurse
Tell the name server to query other servers if it does not have the information. The default is **recurse**.

ret[ry]=*count*
Set the number of times to retry a request before giving up to *count*. When a reply to a request is not received within a certain amount of time (changed with **set timeout**), the request is resent. The default is *count* is **2**.

ro[ot]=*host*
Change the name of the root server to *host*. This affects the **root** command. The default root server is **sri-nic.arpa**.

t[timeout]=*interval*
Change the time-out for a reply to *interval* seconds. The default *interval is* **10** seconds.

[no]v[c]
Always use a virtual circuit when sending requests to the server. The default is **novc**.

DIAGNOSTICS

If the lookup request was not successful, an error message is printed. Possible errors are:

Time-out
The server did not respond to a request after a certain amount of time (changed with **set** *timeout*=**value**) and a certain number of retries (changed with **set** *retry*=**value**).

No information
Depending on the query type set with the **set querytype** command, no information about the host was available, though the host name is valid.

Non-existent domain
The host or domain name does not exist.

Connection refused
Network is unreachable
The connection to the name or finger server could not be made at the current time. This error commonly occurs with **finger** requests.

Server failure
The name server found an internal inconsistency in its database and could not return a valid answer.

Refused
The name server refused to service the request.

The following error should not occur and it indicates a bug in the program.

Format error
The name server found that the request packet was not in the proper format.

FILES

/etc/resolv.conf initial domain name and name server addresses.

SEE ALSO

resolver(3), **resolv.conf**(5), **named**(8C)

RFC 1034, RFC 1035

System and Network Administration

NAME

nsquery – RFS name server query

SYNOPSIS

nsquery [**–h**] [*name*]

AVAILABILITY

This program is available with the *RFS* software installation option. Refer to *Installing SunOS 4.1* for information on how to install optional software.

DESCRIPTION

nsquery provides information about resources available to the host from both the local domain and from other domains. All resources are reported, regardless of whether the host is authorized to access them. When used with no options, **nsquery** identifies all resources in the domain that have been advertised as sharable. A report on selected resources can be obtained by specifying *name*, where *name* is one of:

nodename	The report will include only those resources available from *nodename*.
domain.	The report will include only those resources available from *domain*.
domain.nodename	The report will include only those resources available from *domain.nodename*.

When the name does not include the delimiter '.', it will be interpreted as a *nodename* within the local domain. If the name ends with a delimiter '.', it will be interpreted as a domain name.

The information contained in the report on each resource includes its advertised name (*domain.resource*), the read/write permissions, the server (*nodename.domain*) that advertised the resource, and a brief textual description.

A remote domain must be listed in your **rfmaster** file in order to query that domain.

If no entries are found when **nsquery** is executed, the report header is printed.

If your host cannot contact the domain name server, an error message will be sent to standard error.

OPTIONS

–h Do not print header.

EXAMPLE

The following example displays the resources available from the domain **sunrfs**:

```
example% nsquery sunrfs.
RESOURCE        ACCESS      SERVER          DESCRIPTION

LOCAL_SUN3      read-only   sunrfs.estale
LOCAL_SUN4      read-only   sunrfs.estale
LOCAL_SHARE     read-only   sunrfs.estale
```

SEE ALSO

rfmaster(5), **adv**(8), **unadv**(8)

NAME

old-analyze, analyze – postmortem system crash analyzer

SYNOPSIS

/usr/old/analyze [**–dfmvD**] [**–s** *swapfile*] *corefile* [**system**]

DESCRIPTION

analyze is the post-mortem analyzer for the state of the paging system. In order to use **analyze** you must arrange to get a image of the memory (and possibly the paging area) of the system after it crashes (see **panic**(8S)).

The **analyze** program reads the relevant system data structures from the core image file and indexing information from **/vmunix** (or the specified file) to determine the state of the paging subsystem at the point of crash. It looks at each process in the system, and the resources each is using in an attempt to determine inconsistencies in the paging system state. Normally, the output consists of a sequence of lines showing each active process, its state (whether swapped in or not), its *p0br*, and the number and location of its page table pages. Any pages which are locked while raw I/O is in progress, or which are locked because they are *intransit* are also printed. (Intransit text pages often diagnose as duplicated; you will have to weed these out by hand.)

The program checks that any pages in core which are marked as not modified are, in fact, identical to the swap space copies. It also checks for non-overlap of the swap space, and that the core map entries correspond to the page tables. The state of the free list is also checked.

Options to **analyze**:

–d Print the (sorted) paging area usage.

–f Dump the free list.

–m Dump the entire coremap state.

–v (Long unused.) Use a hugely verbose output format.

–D Print the diskmap for each process.

In general, the output from this program can be confused by processes which were forking, swapping, or exiting or happened to be in unusual states when the crash occurred. You should examine the flags fields of relevant processes in the output of a **pstat**(8) to weed out such processes.

It is possible to look at the core dump with **adb**(1) if you do

adb –k /vmunix /vmcore

FILES

/vmunix default system namelist

SEE ALSO

adb(1), **ps**(1), **panic**(8S), **pstat**(8)

DIAGNOSTICS

Various diagnostics about overlaps in swap mappings, missing swap mappings, page table entries inconsistent with the core map, incore pages which are marked clean but differ from disk-image copies, pages which are locked or intransit, and inconsistencies in the free list.

It would be nice if this program analyzed the system in general, rather than just the paging system in particular.

NAME

pac – printer/plotter accounting information

SYNOPSIS

/usr/etc/pac [**–cmrs**] [**–P***printer*] [**–p***price*] [*username*...]

DESCRIPTION

pac reads the printer/plotter accounting files, accumulating the number of pages (the usual case) or feet (for raster devices) of paper consumed by each user, and printing out how much each user consumed in pages or feet and dollars. The accounting file is taken from the **af** field of the **printcap** entry for the printer. If any *username*s are specified, then statistics are only printed for those users; usually, statistics are printed for every user who has used any paper.

OPTIONS

–c Sort the output by cost; usually the output is sorted alphabetically by name.

–m Disregard machine names. Normally, print jobs submitted by a user from different machines would be counted separately for each machine.

–r Reverse the sorting order.

–s Summarize the accounting information on the summary accounting file. The name of the summary file is the name of the accounting file with '**_sum**' appended to it.

–P*printer* Do accounting for the named *printer*. If this option is not used, the printer specified by the PRINTER environment variable will be used if it is present; otherwise accounting is done for the default printer.

–p*price* Use the value *price* for the cost in dollars per page/foot instead of the default value of 0.02.

FILES

/etc/printcap

SEE ALSO

printcap(5)

BUGS

The relationship between the computed price and reality is as yet unknown.

NAME

panic – what happens when the system crashes

DESCRIPTION

This section explains what happens when the system crashes and how you can analyze crash dumps.

When the system crashes voluntarily, it displays a message of the form

panic: *why i gave up the ghost*

on the console, takes a dump on a mass storage peripheral, and then invokes an automatic reboot procedure as described in **reboot**(8). Unless some unexpected inconsistency is encountered in the state of the file systems due to hardware or software failure, the system will then resume multiuser operations.

The system has a large number of internal consistency checks; if one of these fails, it will panic with a very short message indicating which one failed.

When the system crashes it writes (or at least attempts to write) an image of memory into the back end of the primary swap area. After the system is rebooted, you can run the program **savecore**(8) to preserve a copy of this core image and kernel namelist for later perusal. See **savecore**(8) for details.

To analyze a dump you should begin by running **adb**(1) with the **–k** flag on the core dump, as described in *Debugging Tools*.

The most common cause of system failures is hardware failure, which can reflect itself in different ways.

See DIAGNOSTICS for some messages that you may encounter, with some hints as to causes. In each case there is a possibility that a hardware or software error produced the message in some unexpected way.

FILES

/vmunix	the system kernel
/etc/rc.local	script run when the local system starts up

SEE ALSO

adb(1), **old-analyze**(8), **reboot**(8) **sa**(8), **savecore**(8)

Debugging Tools

DIAGNOSTICS

IO err in push

hard IO err in swap The system encountered an error trying to write to the paging device or an error in reading critical information from a disk drive. You should fix your disk if it is broken or unreliable.

timeout table overflow

This really should not be a panic, but until the data structure is fixed, involved, running out of entries causes a crash. If this happens, you should make the timeout table bigger by changing the value of **ncallout** in the **param.c** file, and then rebuild your system.

trap type *type*, **pid** *process-id*, **pc** = *program-counter*, **sr** = *status-register*, **context** *context-number*

A unexpected trap has occurred within the system; typical trap types are:

- Bus error
- Address error
- Illegal instruction
- Divide by zero
- Chk instruction
- Trapv instruction
- Privilege violation
- Trace
- 1010 emulator trap
- 1111 emulator trap
- Stack format error

- Uninitialized interrupt
- Spurious interrupt

The favorite trap types in system crashes are "Bus error" or "Address error", indicating a wild reference. The *process-id* is the ID of the process running at the time of the fault, *program-counter* is the hexadecimal value of the program counter, *status-register* is the hexadecimal value of the status register, and *context-number* is the context that the process was running in. These problems tend to be easy to track down if they are kernel bugs since the processor stops cold, but random flakiness seems to cause this sometimes.

init died

The system initialization process has exited. This is bad news, as no new users will then be able to log in. Rebooting is the only fix, so the system just does it right away.

NAME

ping – send ICMP ECHO_REQUEST packets to network hosts

SYNOPSIS

/usr/etc/ping *host* [*timeout*]

/usr/etc/ping [**–s**] [**–lrRv**] *host* [*packetsize*] [*count*]

AVAILABILITY

This program is available with the *Networking* software installation option. Refer to *Installing SunOS 4.1* for information on how to install optional software.

DESCRIPTION

ping utilizes the ICMP protocol's mandatory ECHO_REQUEST datagram to elicit an ICMP ECHO_RESPONSE from the specified *host* , or network gateway. ECHO_REQUEST datagrams, or "pings," have an IP and ICMP header, followed by a *struct*timeval, and then an arbitrary number of bytes to pad out the packet. If *host* responds, **ping** will print *host* **is alive** on the standard output and exit. Otherwise after *timeout* seconds, it will write **no answer from** *host*. The default value of *timeout* is 20 seconds.

When the **–s** flag is specified, **ping** sends one datagram per second, and prints one line of output for every ECHO_RESPONSE that it receives. No output is produced if there is no response. In this second form, **ping** computes round trip times and packet loss statistics; it displays a summary of this information upon termination or timeout. The default datagram packet size is 64 bytes, or you can specify a size with the *packetsize* command-line argument. If an optional *count* is given, **ping** sends only that number of requests.

When using **ping** for fault isolation, first '**ping**' the local host to verify that the local network interface is running.

OPTIONS

–l Loose source route. Use this option in the IP header to send the packet to the given host and back again. Usually specified with the **–R** option.

–r Bypass the normal routing tables and send directly to a host on an attached network. If the host is not on a directly-attached network, an error is returned. This option can be used to **ping** a local host through an interface that has been dropped by the router daemon, see **routed**(8C).

–R Record route. Sets the IP record route option, which will store the route of the packet inside the IP header. The contents of the record route will only be printed if the –v option is given, and only be set on return packets if the target host preserves the record route option across echos, or the **–l** option is given.

–v Verbose output. List any ICMP packets, other than ECHO_RESPONSE, that are received.

SEE ALSO

icmp(4P), **ifconfig**(8C), **netstat**(8C), **rpcinfo**(8C), **spray**(8C)

NAME

pnpboot, pnp.s386 – pnp diskless boot service

SYNOPSIS

/tftpboot/pnp.s386

AVAILABILITY

Available only on Sun 386i systems running a SunOS 4.0.*x* release or earlier. Not a SunOS 4.1 release feature.

DESCRIPTION

pnp.s386 is a level 2 boot program that requests actions necessary to set up a diskless workstation on the network.

The PNP diskless boot service is used by diskless workstations at installation time to locate a server that will configure the diskless client.

The last steps of the level 1 boot (from the PROM) are to load the level 2 program through **rarpd**(8C) and **tftpd**(8C). The first step in the boot sequence is RARP to acquire an IP address. This is followed by TFTP service calls to acquire the **pnp.sun*** program file needed for the client's architecture. A **PNP_ACQUIRE** RPC is then broadcast to locate a server willing to configure the diskless client.

A **PNP_SETUP** is issued to the server which returns one of three statuses: success, failure, or in_progress. As long as the server responds with a status of in_progress the client will periodically issue a **PNP_POLL** until the status changes to either success or failure.

The last step is to reboot the client. This goes through a RARP, TFTP, BOOT sequence, with the boot using the normal **boot.sun*** file and **bootparamd**(8) service.

The system will have been set up using the IP address returned in the first step and a system name will have been assigned.

FILES

/tftpboot/pnp.sun*

SEE ALSO

bootparam(3R), **bootparams**(5) **boot**(8S), **bootparamd**(8), **ipallocd**(8C), **netconfig**(8C), **pnpd**(8C), **rarpd**(8C), **tftpd**(8C)

NAME

pnpd – PNP daemon

SYNOPSIS

/usr/etc/rpc.pnpd

AVAILABILITY

Available only on Sun 386i systems running a SunOS 4.0*x* release or earlier. Not a SunOS 4.1 release feature.

DESCRIPTION

pnpd is used during routine booting of systems to determine their network configuration, and by new systems to configure themselves on a network. **pnpd** adds and removes diskless clients of the boot server on which it is running. The **pnpd** daemon is normally invoked in **rc.local**. The RPCs are used by **netconfig**(8C), **pnp.s386** (see **pnpboot**(8C)), and **client**(8).

The **bootservers** Network Interface Service (NIS) map specifies limits on server capacity and default swap size.

FILES

/export/exec/*arch*
: symbolic link to **/export/exec/***arch.release*

/export/exec/*arch.release*
: symbolic link to **/usr** for the architecture

/export/exec/*arch.release***/boot**
: root binaries

SEE ALSO

pnp(3R), **client**(8), **ipallocd**(8C), **netconfig**(8C), **pnpboot**(8C)

NOTES

The Network Information Service (NIS) was formerly known as Sun Yellow Pages (YP). The functionality of the two remains the same; only the name has changed.

NAME

portmap – TCP/IP port to RPC program number mapper

SYNOPSIS

/usr/etc/portmap

DESCRIPTION

portmap is a server that converts TCP/IP protocol port numbers into RPC program numbers. It must be running in order to make RPC calls.

When an RPC server is started, it will tell **portmap** what port number it is listening to, and what RPC program numbers it is prepared to serve. When a client wishes to make an RPC call to a given program number, it will first contact **portmap** on the server machine to determine the port number where RPC packets should be sent.

Normally, standard RPC servers are started by **inetd**(8C), so **portmap** must be started before **inetd** is invoked.

SEE ALSO

inetd.conf(5), **inetd**(8C), **rpcinfo**(8C)

BUGS

If **portmap** crashes, all servers must be restarted.

NAME

praudit – print contents of an audit trail file

SYNOPSIS

praudit [**–lrs**] [**–d***del*] [*filename* ...]

AVAILABILITY

This program is available with the *Security* software installation option. Refer to *Installing SunOS 4.1* for information on how to install optional software.

DESCRIPTION

praudit reads the listed *filename*s (or standard input, if no *filename* is specified) and interprets the data as audit trail records as defined in **audit_control**(5). By default, times, security labels, user and group IDs (UIDs and GIDs, respectively) are converted to their ASCII representation. Record type and event fields are converted to long ASCII representation. A maximum of 100 audit files can be specified on the command line.

OPTIONS

–l Print records one line per record. The record type and event fields are always converted to their short ASCII representation.

–r Print records in their raw form. Times, security labels, UIDs, GIDs, record types, and events are displayed as integers. Currently, labels are not used and are displayed as zero in this mode. This option and the –s option are exclusive. If both are used, a format usage error message is output.

–s Print records in their short form. All numeric fields are converted to ASCII and displayed. The short ASCII representations for the record type and event fields are used. Security labels are displayed in their short representation. Again, labels are not currently used. This option and the –r option are exclusive. If both are used, a format usage error message is output.

–d*del* Use *del* as the field delimiter instead of the default delimiter, which is the comma. If *del* has special meaning for the shell, it must be quoted. The maximum size of a delimiter is four characters.

FILES

/etc/passwd

SEE ALSO

audit(2), **setuseraudit**(2), **getauditflags**(3), **audit_control**(5)

NAME

pstat – print system facts

SYNOPSIS

/usr/etc/pstat [**–afipSsT**] [**–u** *pid*] [*system* [*corefile*]]

DESCRIPTION

pstat interprets the contents of certain system tables. If *corefile* is given, the tables are sought there, otherwise in **/dev/kmem**. The required namelist is taken from **/vmunix** unless *system* is specified.

OPTIONS

–a Under **–p**, describe all process slots rather than just active ones.

–f Print the open file table with these headings:

LOC	The memory address of this table entry.
TYPE	The type of object the file table entry points to.
FLG	Miscellaneous state variables encoded thus: R open for reading W open for writing A open for appending S shared lock present X exclusive lock present I signal pgrp when data ready
CNT	Number of processes that know this open file.
MSG	Number of references from message queue.
DATA	The location of the vnode table entry or socket for this file.
OFFSET	The file offset (see **lseek**(2V)).

–i Print the inode table including the associated vnode entries with these headings:

ILOC	The memory address of this table entry.
IFLAG	Miscellaneous inode state variables encoded thus: A inode access time must be corrected C inode change time must be corrected L inode is locked R inode is being referenced U update time (fs(5)) must be corrected W wanted by another process (L flag is on)
IDEVICE	Major and minor device number of file system in which this inode resides.
INO	I-number within the device.
MODE	Mode bits in octal, see **chmod**(2V).
NLK	Number of links to this inode.
UID	User ID of owner.
SIZE/DEV	Number of bytes in an ordinary file, or major and minor device of special file.
VFLAG	Miscellaneous vnode state variables encoded thus: R root of its file system S shared lock applied E exclusive lock applied Z process is waiting for a shared or exclusive lock
CNT	Number of open file table entries for this vnode.
SHC	Reference count of shared locks on the vnode.
EXC	Reference count of exclusive locks on the vnode (this may be '> 1' if, for example, a file descriptor is inherited across a fork).
TYPE	Vnode file type, either VNON (no type), VREG (regular), VDIR (directory), VBLK (block device), VCHR (character device), VLNK (symbolic link), VSOCK (socket), VFIFO (named pipe), or VBAD (bad).

-p Print process table for active processes with these headings:

LOC — The memory address of this table entry.

S — Run state encoded thus:

0 no process
1 awaiting an event
2 (abandoned state)
3 runnable
4 being created
5 being terminated
6 stopped (by signal or under trace)

F — Miscellaneous state variables, ORed together (hexadecimal):

0000001	loaded
0000002	a system process (scheduler or page-out daemon)
0000004	locked for swap out
0000008	swapped out during process creation
0000010	process is being traced
0000020	tracing parent has been told that process is stopped
0000040	user settable lock in memory
0000080	in page-wait
0000100	prevented from swapping during **fork**(2V)
0000200	will restore old mask after taking signal
0000400	exiting
0000800	doing physical I/O
0001000	process resulted from a **vfork**(2) which is not yet complete
0002000	another flag for **vfork**(2)
0004000	process has no virtual memory, as it is a parent in the context of **vfork**(2)
0008000	process is demand paging pages from its executable image vnode
0010000	process has advised of sequential VM behavior with **vadvise**(2)
0020000	process has advised of random VM behavior with **vadvise**(2)
0080000	process is a session process group leader
0100000	process is tracing another process
0200000	process needs a profiling tick
0400000	process is scanning descriptors during select
4000000	process has done record locks
8000000	process is having its system calls traced

PRI — Scheduling priority, see **getpriority**(2).

SIG — Signals received (signals 1-32 coded in bits 0-31).

UID — Real user ID.

SLP — Amount of time process has been blocked.

TIM — Time resident in seconds; times over 127 coded as 127.

CPU — Weighted integral of CPU time, for scheduler.

NI — Nice level, see **getpriority**(2).

PGRP — Process number of root of process group.

PID — The process ID number.

PPID — The process ID of parent process.

RSS — Resident set size — the number of physical page frames allocated to this process.

SRSS — RSS at last swap (0 if never swapped).

SIZE The size of the process image. That is, the sum of the data and stack segment sizes, not including the sizes of any shared libraries.
WCHAN Wait channel number of a waiting process.
LINK Link pointer in list of runnable processes.

-S Print the streams table with these headings:

LOC The memory address of this table entry.
WRQ The address of this stream's write queue.
VNODE The address of this stream's vnode.
DEVICE Major and minor device number of device to which this stream refers.
PGRP This stream's process group number.
SIGIO The process id or process group that has this stream **open()**.
FLG Miscellaneous stream state variables encoded thus:

- I waiting for **ioctl()** to finish
- R read/recvmsg is blocked
- W write/putmsg is blocked
- P priority message is at stream head
- H device has been "hung up" (**M_HANGUP**)
- O waiting for open to finish
- M stream is linked under multiplexor
- D stream is in message-discard mode
- N stream is in message-nondiscard mode
- E fatal error has occurred (**M_ERROR**)
- T waiting for queue to drain when closing
- 2 waiting for previous **ioctl()** to finish before starting new one
- 3 waiting for acknowledgment for **ioctl()**
- B stream is in non-blocking mode
- A stream is in asynchronous mode
- o stream uses old-style no-delay mode
- S stream has had **TOSTOP** set
- C VTIME clock running
- V VTIME timer expired
- r collision on **select()** for reading
- w collision on **select()** for writing
- e collision on **select()** for exceptional condition

The queues on the write and read sides of the stream are listed for each stream. Each queue is printed with these headings:

NAME The name of the module or driver for this queue.
COUNT The approximate number of bytes on this queue.
FLG Miscellaneous state variables encoded thus:

- E queue is enabled to run
- R someone wants to get from this queue when it becomes non-empty
- W someone wants to put on this queue when it drains
- F queue is full
- N queue should not be enabled automatically by a putq

MINPS The minimum packet size for this queue.
MAXPS The maximum packet size for this queue, or **INF** if there is no maximum.
HIWAT The high-water mark for this queue.
LOWAT The low-water mark for this queue.

-s Print information about swap space usage:

allocated: The amount of swap space (in bytes) allocated to private pages.

reserved: The number of swap space bytes not currently allocated, but claimed by memory mappings that have not yet created private pages.

used: The total amount of swap space, in bytes, that is either allocated or reserved.

available: The total swap space, in bytes, that is currently available for future reservation and allocation.

-T Print the number of used and free slots in the several system tables. This is useful for checking to see how full system tables have become if the system is under heavy load. Shows both used and cached inodes.

-u *pid* Print information about the process with ID *pid*.

FILES

/vmunix	namelist
/dev/kmem	default source of tables

SEE ALSO

ps(1), **chmod**(2V), **fork**(2V), **getpriority**(2), **lseek**(2V), **stat**(2V), **vadvise**(2), **vfork**(2), **fs**(5) **iostat**(8), **vmstat**(8)

BUGS

It would be very useful if the system recorded "maximum occupancy" on the tables reported by **-T**; even more useful if these tables were dynamically allocated.

NAME

pwck – check password database entries

SYNOPSIS

/usr/etc/pwck [*filename*]

AVAILABILITY

This command is available with the *System V* software installation option. Refer to *Installing SunOS 4.1* for information on how to install optional software.

DESCRIPTION

pwck checks that a file in **passwd**(5) does not contain any errors; it checks the **/etc/passwd** file by default.

FILES

/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 lower-case letters and digits.

First char in login name not lower case alpha
The login name in an entry does not begin with a lower-case 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 Network Interface Service (NIS) entry referring to a netgroup, but no netgroup is present.

Bad character(s) in netgroup name
The netgroup name in an NIS entry contains characters other than lower-case letters and digits.

First char in netgroup name not lower case alpha
The netgroup name in an NIS entry does not begin with a lower-case letter.

SEE ALSO

group(5), **passwd**(5)

NOTES

The Network Information Service (NIS) was formerly known as Sun Yellow Pages (YP). The functionality of the two remains the same; only the name has changed.

NAME

pwdauthd – server for authenticating passwords

SYNOPSIS

/usr/etc/rpc.pwdauthd

AVAILABILITY

This program is available with the *Security* software installation option. Refer to *Installing SunOS 4.1* for information on how to install optional software.

DESCRIPTION

pwdauthd is a server that determines authentication for users and groups. It handles authentication requests from **pwdauth**(3) and **grpauth**( ). Communication to and from **pwdauthd** is by means of RPC calls. The server is passed a *filename* and a *password*. It returns an integer value that specifies whether the *password* is valid. The possible return values are **PWA_VALID** if the name is valid, **PWA_INVALID** if the name is invalid, and **PWA_UNKNOWN** if validity cannot be determined because no adjunct files are present.

If **pwdauthd** is serving *pwdauth*, it determines whether the **passwd.adjunct** file exists. If not, it returns **PWA_UNKNOWN**. In this case, *pwdauth* knows to check the **/etc/passwd** file. Otherwise, the server calls **getpwanam**( ) (see **getpwaent**(3)) to get the entry for *filename* in either the local or the Network Interface Service (NIS) file for **passwd.adjunct**. If the encrypted password guess matches the encrypted password from the file, **pwdauthd** returns **PWA_VALID**. If the passwords do not match, it returns **PWA_INVALID**.

If **pwdauthd** is serving **grpauth**( ), it determines whether the **group.adjunct** file exists. If not, it returns **PWA_UNKNOWN**. In this case, **grpauth**( ) knows to check the **/etc/group** file. Otherwise, the server calls **getgranam**( ) (see **getgraent**(3)) to get the entry for *filename* in either the local or the NIS file for **group.adjunct**. If the encrypted password guess matches the encrypted password from the file, **pwdauthd** returns **PWA_VALID**. If the passwords do not match, it returns **PWA_INVALID**.

SEE ALSO

getgraent(3), **getpwaent**(3), **pwdauth**(3)

NOTES

The Network Information Service (NIS) was formerly known as Sun Yellow Pages (YP). The functionality of the two remains the same; only the name has changed.

NAME

quot – summarize file system ownership

SYNOPSIS

/usr/etc/quot [**–acfhnv**] [*filesystem*]

DESCRIPTION

quot displays the number of blocks (1024 bytes) in the named *filesystem* currently owned by each user.

OPTIONS

–a Generate a report for all mounted file systems.

–c Display three columns giving file size in blocks, number of files of that size, and cumulative total of blocks in that size or smaller file.

–f Display count of number of files as well as space owned by each user.

–h Estimate the number of blocks in the file — this doesn't account for files with holes in them.

–n Run the pipeline **ncheck filesystem | sort +0n | quot –n filesystem** to produce a list of all files and their owners.

–v Display three columns containing the number of blocks not accessed in the last 30, 60, and 90 days.

FILES

/etc/mtab	mounted file systems
/etc/passwd	to get user names

SEE ALSO

du(1V), **ls**(1V)

NAME

quotacheck – file system quota consistency checker

SYNOPSIS

/usr/etc/quotacheck [**–v**] [**–p**] *filesystem*...

/usr/etc/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 is normally run at boot time from the **/etc/rc.local** file, see **rc**(8), before enabling disk quotas with **quotaon**(8).

quotacheck accesses the raw device in calculating the actual disk usage for each user. Thus, the file systems checked should be quiescent while **quotacheck** is running.

OPTIONS

–v Indicate the calculated disk quotas for each user on a particular file system. **quotacheck** normally reports only those quotas modified.

–a Check all the file systems indicated in **/etc/fstab** to be read-write with disk quotas.

–p Run parallel passes on the required file systems, using the pass numbers in **/etc/fstab** in an identical fashion to **fsck**(8).

FILES

quotas	quota file at the file system root
/etc/mtab	mounted file systems
/etc/fstab	default file systems

SEE ALSO

quotactl(2), **quotaon**(8), **rc**(8)

NAME

quotaon, quotaoff – turn file system quotas on and off

SYNOPSIS

/usr/etc/quotaon [**–v**] *filesystem...*
/usr/etc/quotaon [**–av**]

/usr/etc/quotaoff [**–v**] *filesystem...*
/usr/etc/quotaoff [**–av**]

DESCRIPTION

quotaon

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

quotaoff announces to the system that file systems specified should have any disk quotas turned off.

OPTIONS

quotaon

–a All file systems in **/etc/fstab** marked read-write with quotas will have their quotas turned on. This is normally used at boot time to enable quotas.

–v Display a message for each file system where quotas are turned on.

quotaoff

–a Force all file systems in **/etc/fstab** to have their quotas disabled.

–v Display a message for each file system affected.

These commands update the status field of devices located in **/etc/mtab** to indicate when quotas are on or off for each file system.

FILES

quotas	quota file at the file system root
/etc/mtab	mounted file systems
/etc/fstab	default file systems

SEE ALSO

quotactl(2), **fstab**(5), **mtab**(5)

NAME

rarpd – TCP/IP Reverse Address Resolution Protocol server

SYNOPSIS

/usr/etc/rarpd *interface* [*hostname*]

/usr/etc/rarpd –a

AVAILABILITY

This program is available with the *Networking* software installation option. Refer to *Installing SunOS 4.1* for information on how to install optional software.

DESCRIPTION

rarpd starts a daemon that responds to Reverse Address Resolution Protocol (RARP) requests. The daemon forks a copy of itself that runs in background. It must be run as root.

RARP is used by machines at boot time to discover their Internet Protocol (IP) address. The booting machine provides its Ethernet Address in an RARP request message. Using the "ethers" and "hosts" databases, **rarpd** maps this Ethernet Address into the corresponding IP address which it returns to the booting machine in an RARP reply message. The booting machine must be listed in both databases for **rarpd** to locate its IP address. **rarpd** issues no reply when it fails to locate an IP address. The "ethers" and "hosts" databases may be contained either in files under **/etc** or in Network Interface Service (NIS) maps.

In the first synopsis, the *interface* parameter names the network interface upon which **rarpd** is to listen for requests. The *interface* parameter takes the "name unit" form used by **ifconfig**(8C). The second argument, *hostname*, is used to obtain the IP address of that interface. An IP address in "decimal dot" notation may be used for *hostname*. If *hostname* is omitted, the address of the interface will be obtained from the kernel. When the first form of the command is used, **rarpd** must be run separately for each interface on which RARP service is to be supported. A machine that is a router may invoke **rarpd** multiple times, for example:

```
/usr/etc/rarpd ie0 host
/usr/etc/rarpd ie1 host-backbone
```

In the second synopsis, **rarpd** locates all of the network interfaces present on the system and starts a daemon process for each one that supports RARP.

FILES

/etc/ethers
/etc/hosts

SEE ALSO

ethers(5), **hosts**(5), **policies**(5), **boot**(8S), **ifconfig**(8C), **ipallocd**(8C), **netconfig**(8C)

Finlayson, Ross, Timothy Mann, Jeffrey Mogul, and Marvin Theimer, *A Reverse Address Resolution Protocol*, RFC 903, Network Information Center, SRI International, Menlo Park, Calif., June 1984.

NOTES

The Network Information Service (NIS) was formerly known as Sun Yellow Pages (YP). The functionality of the two remains the same; only the name has changed.

NAME

rc, rc.boot, rc.local – command scripts for auto-reboot and daemons

SYNOPSIS

/etc/rc

/etc/rc.boot

/etc/rc.local

DESCRIPTION

rc and **rc.boot** are command scripts that are invoked by **init**(8) to perform file system housekeeping and to start system daemons. **rc.local** is a script for commands that are pertinent only to a specific site or client machine.

rc.boot sets the machine name, and then, if coming up multi-user, runs **fsck**(8) with the **–p** option. This "preens" the disks of minor inconsistencies resulting from the last system shutdown and checks for serious inconsistencies caused by hardware or software failure. If **fsck**(8) detects a serious disk problem, it returns an error and **init**(8) brings the system up in single-user mode. When coming up single-user, when **init**(8) is invoked by **fastboot**(8), or when it is passed the **–b** flag from **boot**(8S), functions performed in the **rc.local** file, including this disk check, are skipped.

Next, **rc** runs. If the system came up single-user, **rc** runs when the single-user shell terminates (see **init**(8)). It mounts 4.2 filesystems and spawns a shell for **/etc/rc.local**, which mounts NFS filesystems, and starts local daemons. After **rc.local** returns, **rc** starts standard daemons, preserves editor files, clears **/tmp**, starts system accounting (if applicable), starts the network (where applicable), and if enabled, runs **savecore**(8) to preserve the core image after a crash.

Sun386i

These files operate as described above with the following variations:

fsck(8) is invoked with the **–y** option to prevent users being put in single-user mode by happenstance.

rc.boot invokes **netconfig**(8C) to configure the system for the network before booting. **netconfig** is invoked before the **/usr** filesystem is mounted, because **/usr** might be mounted from a server. **netconfig** writes **/etc/net.conf** unless the **–n** option is specified, controlling system booting.

rc.boot dynamically loads device drivers.

rc invokes any programs found in **/var/recover** to clean up any operations partially completed when the system crashed or was shut down.

rc.local starts the automounter.

The file **/etc/net.conf** stores these environment variables: The **VERBOSE** environment variable controls the verbosity of the messages from the **rc** script; its value is taken from **NVRAM**. The **NETWORKED** environment variable controls whether services useful only on a networked system are started in **/etc/rc.local**. The **PNP** environment variable, set up during initial system installation, controls whether local network configuration information is used or whether that information comes from the network. (Using automatic system installation causes all systems except boot servers to get this information from the network, facilitating network reconfiguration.) The **HOSTNAME** and **DOMAINNAME** environment variables, used together, help determine if this system is a boot server or, with **PNP** set to **no**, control the host name and domain name.

FILES

/etc/rc
/etc/rc.boot
/etc/rc.local
/etc/net.conf
/var/recover/*
/var/yp/*
/tmp

SEE ALSO

automount(8), **boot**(8S), **fastboot**(8), **init**(8), **reboot**(8), **savecore**(8), **netconfig**(8C)

BUGS

The system message file **/var/adm/messages** is no longer created automatically.

NAME

rdate – set system date from a remote host

SYNOPSIS

/usr/ucb/rdate *hostname*

AVAILABILITY

This program is available with the *Networking* software installation option. Refer to *Installing SunOS 4.1* for information on how to install optional software.

DESCRIPTION

rdate sets the local date and time from the *hostname* given as argument. You must be super-user on the local system. Typically **rdate** can be inserted as part of your **/etc/rc.local** startup script.

FILES

/etc/rc.local

BUGS

Could be modified to accept a list of hostnames and try each until a valid date returned. Better yet would be to write a real date server that accepted broadcast requests.

NAME

reboot – restart the operating system

SYNOPSIS

/usr/etc/reboot [**–dnq**] [*boot arguments*]

DESCRIPTION

reboot executes the **reboot**(2) system call to restart the kernel. The kernel is loaded into memory by the PROM monitor, which transfers control to it. See **boot**(8S) for details.

Although **reboot** can be run by the super-user at any time, **shutdown**(8) is normally used first to warn all users logged in of the impending loss of service. See **shutdown**(8) for details.

reboot performs a **sync**(1) operation on the disks, and then a multiuser reboot is initiated. See **init**(8) for details.

reboot normally logs the reboot to the system log daemon, **syslogd**(8), 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.

Power Fail and Crash Recovery

Normally, the system will reboot itself at power-up or after crashes.

OPTIONS

–d Dump system core before rebooting.

–n Avoid the **sync**(1). 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

If a boot argument string is given, it is passed to the boot command in the PROM monitor. The string must be quoted if it contains spaces or other characters that could be interpreted by the shell. If the first character of the boot argument string is a minus sign '–' the string must be preceded by an option terminator string '––' For example: 'reboot – – –s' to reboot and come up single user, '**reboot vmunix.test**' to reboot to a new kernel. See **boot**(8S) for details.

FILES

/var/adm/wtmp login accounting file

SEE ALSO

sync(1), **reboot**(2), **boot**(8S), **fsck**(8), **halt**(8), **init**(8), **panic**(8S), **shutdown**(8), **syslogd**(8)

NAME

renice – alter nice value of running processes

SYNOPSIS

/usr/etc/renice *priority pid...*

/usr/etc/renice *priority* [**–p** *pid...*] [**–g** *pgrp...*] [**–u** *username...*]

DESCRIPTION

renice alters the scheduling nice value, and hence the priority, of one or more running processes. See **nice**(1) for a discussion of nice value and process scheduling priority.

OPTIONS

By default, the processes to be affected are specified by their process IDs. *priority* is the new priority value.

–p *pid ...* Specify a list of process IDs.

–g *pgrp ...* Specify a list of process group IDs. The processes in the specified process groups have their scheduling priority altered.

–u *user ...* Specify a list of user IDs or usernames. All processes owned by each *user* have their scheduling altered.

Users other than the super-user may only alter the priority of processes they own, and can only monotonically increase their "nice value" within the range 0 to 20. (This prevents overriding administrative fiats.) The super-user may alter the priority of any process and set the priority to any value in the range –20 to 19. Useful nice values are 19 (the affected processes will run only when nothing else in the system wants to), 0 (the default nice value) and any negative value (to make things go faster).

If only the priority is specified, the current process (alternatively, process group or user) is used.

FILES

/etc/passwd to map user names to user ID's

SEE ALSO

pstat(8)

BUGS

If you make the nice value very negative, then the process cannot be interrupted.

To regain control you must make the priority greater than zero.

Users other than the super-user cannot increase scheduling priorities of their own processes, even if they were the ones that decreased the priorities in the first place.

NAME

repquota – summarize quotas for a file system

SYNOPSIS

/usr/etc/repquota [**–v**] *filesystem*...

/usr/etc/repquota [**–av**]

DESCRIPTION

repquota prints a summary of the disc 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**(8).

OPTIONS

–a Report on all file systems indicated in **/etc/fstab** to be read-write with quotas.

–v Report all quotas, even if there is no usage.

Only the super-user may view quotas which are not their own.

FILES

quotas	quota file at the file system root
/etc/fstab	default file systems

SEE ALSO

quota(1), **quotactl**(2), **edquota**(8), **quotacheck**(8), **quotaon**(8)

NAME

restore, rrestore – incremental file system restore

SYNOPSIS

/usr/etc/restore –irRtx [*filename* ...]

DESCRIPTION

restore restores files from backup tapes created with the **dump**(8) 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 **restore** 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, **restore** 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 Restore the entire tape. Load the tape's full contents into the current directory. This option should only be used to restore a complete dump tape onto a clear filesystem, or to restore an incremental dump tape after a full "level **0**" restore. For example:

```
example# /usr/etc/newfs /dev/rxy0g
example# /usr/etc/mount /dev/xy0g /mnt
example# cd /mnt
example# restore r
```

is a typical sequence to restore a "level **0**" dump. Another **restore** can be done to get an incremental dump in on top of this.

R Resume restoring. **restore** requests a particular tape of a multivolume set from which to resume a full restore (see the **r** option above). This allows **restore** to start from a checkpoint when it is interrupted in the middle of a full restore.

t Table of contents. List 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. (The **t** option replaces the function of the old **dumpdir** program).

x Extract 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.

Modifiers

Some of the following modifiers take arguments that are given as separate words on the command line. When more than one such modifier appears within *options*, the arguments must appear in the same order as the modifiers that they apply to.

a *archive-file*
The dump table of contents is taken from the specified *archive-file* instead of from a dump tape. If a requested file is present in the table of contents, *restore* will prompt for the tape volume to be mounted. If only contents information is needed, for example when the *t* option is specified, or the *i* option is specified without a corresponding *extract* request, no dump tape will have to be mounted.

c Convert the contents of the dump tape to the new filesystem format.

d Debug. Turn on debugging output.

h Extract the actual directory, rather than the files that it references. This prevents hierarchical restoration of complete subtrees from the tape.

m Extract 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. **restore** displays the name of each file it restores, preceded by its file type.

y Do not ask whether to abort the restore in the event of tape errors. **restore** tries to skip over the bad tape block(s) and continue as best it can.

b *factor*
Blocking factor. Specify the blocking factor for tape reads. By default, **restore** will attempt to figure out the block size of the tape. Note: a tape block is 512 bytes.

f *dump-file*
Use *dump-file* instead of **/dev/rmt?** as the file to restore from. If *dump-file* is specified as '–', **restore** reads from the standard input. This allows, **dump**(8) and **restore** to be used in a pipeline to dump and restore a file system:

example# dump 0f – /dev/rxy0g | (cd /mnt; restore xf –)

If the name of the file is of the form *machine*:*device* the restore is done from the specified machine over the network using **rmt**(8C). Since **restore** is normally run by root, the name of the local machine must appear in the **.rhosts** file of the remote machine. If the file is specified as *user*@*machine*:*device*, **restore** will attempt to execute as the specified user on the remote machine. The specified user must have a **.rhosts** file on the remote machine that allows root from the local machine.

s *n* Skip to the *n*'th file when there are multiple dump files on the same tape. For example, the command:

example# restore xfs /dev/nrar0 5

would position you at the fifth file on the tape.

USAGE

Interactive Commands

restore 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 '/' (slash). 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. **restore** 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.

verbose Toggle the status of the **v** modifier. While **v** is in effect, the **ls** command lists the inode numbers of all entries, and **restore** displays information about each file as it is extracted.

help Display a summary of the available commands.

quit **restore** exits immediately, even if the extraction list is not empty.

FILES

/dev/rmt8	the default tape drive
dumphost:/dev/rmt8	the default tape drive if called as **rrestore**
/tmp/rstdir*	file containing directories on the tape
/tmp/rstmode*	owner, mode, and timestamps for directories
./restoresymtable	information passed between incremental restores

SEE ALSO

dump(8), **mkfs**(8), **mount**(8), **newfs**(8), **rmt**(8C)

DIAGNOSTICS

restore complains about bad option characters.

Read errors result in complaints. If **y** has been specified, or the user responds **y**, **restore** will attempt to continue.

If the dump extends over more than one tape, **restore** asks the user to change tapes. If the **x** or **i** option has been specified, **restore** also asks which volume the user wishes to mount.

There are numerous consistency checks that can be listed by **restore**. 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 restore, skipped *num* blocks

After a tape read error, **restore** may have to resynchronize itself. This message lists the number of blocks that were skipped over.

BUGS

restore 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 **restore** runs in user mode, it has no control over inode allocation; this means that **restore** 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.

NAME

rexd, rpc.rexd – RPC-based remote execution server

SYNOPSIS

/usr/etc/rpc.rexd [–s]

DESCRIPTION

rexd is the Sun RPC server for remote program execution. This daemon is started by **inetd**(8C) whenever a remote execution request is made.

For noninteractive programs, the standard file descriptors are connected directly to TCP connections. Interactive programs involve pseudo-terminals, in a fashion that is similar to the login sessions provided by **rlogin**(1C). This daemon may use NFS to mount file systems specified in the remote execution request.

FILES

/dev/ttyp*n*	pseudo-terminals used for interactive mode
/etc/passwd	authorized users
/tmp_rex/rexd??????	temporary mount points for remote file systems.

OPTIONS

–s Secure. When specified, requests must have valid des credentials. If the request does not have a DES credential it is rejected. The default publickey credential is rejected. Only newer **on** commands send DES credentials.

If access is denied with an Authentication error, you may have to set your publickey with the **chkey**(1) command.

SEE ALSO

chkey(1), **on**(1C), **rlogin**(1C), **rex**(3R), **exports**(5), **inetd.conf**(5), **publickey**(5), **inetd**(8C)

DIAGNOSTICS

Diagnostic messages are normally printed on the console, and returned to the requestor.

RESTRICTIONS

Root cannot execute commands using **rexd** client programs such as **on**(1C).

NAME

rexecd, in.rexecd – remote execution server

SYNOPSIS

/usr/etc/in.rexecd *host.port*

AVAILABILITY

This program is available with the *Networking* software installation option. Refer to *Installing SunOS 4.1* for information on how to install optional software.

DESCRIPTION

rexecd is the server for the **rexec**(3N) routine. The server provides remote execution facilities with authentication based on user names and encrypted passwords. It is invoked automatically as needed by **inetd**(8C), and then executes the following protocol:

- The server reads characters from the socket up to a null (\0) byte. The resultant string is interpreted as an ASCII number, base 10.
- If the number received in step 1 is non-zero, it is interpreted as the port number of a secondary stream to be used for the **stderr**. A second connection is then created to the specified port on the client's machine.
- A null terminated user name of at most 16 characters is retrieved on the initial socket.
- A null terminated, encrypted, password of at most 16 characters is retrieved on the initial socket.
- A null terminated command to be passed to a shell is retrieved on the initial socket. The length of the command is limited by the upper bound on the size of the system's argument list.
- **rexecd** then validates the user as is done at login time and, if the authentication was successful, changes to the user's home directory, and establishes the user and group protections of the user. If any of these steps fail the connection is aborted with a diagnostic message returned.
- A null byte is returned on the connection associated with the **stderr** and the command line is passed to the normal login shell of the user. The shell inherits the network connections established by **rexecd**.

SEE ALSO

rexec(3N) **inetd**(8C)

DIAGNOSTICS

All diagnostic messages are returned on the connection associated with the **stderr**, after which any network connections are closed. An error is indicated by a leading byte with a value of 1 (0 is returned in step 7 above upon successful completion of all the steps prior to the command execution).

username too long
The name is longer than 16 characters.

password too long
The password is longer than 16 characters.

command too long
The command line passed exceeds the size of the argument list (as configured into the system).

Login incorrect.
No password file entry for the user name existed.

Password incorrect.
The wrong password was supplied.

No remote directory.
The **chdir** command to the home directory failed.

Try again.
A **fork** by the server failed.

/usr/bin/sh: ...
The user's login shell could not be started.

BUGS

Indicating **'Login incorrect'** as opposed to **'Password incorrect'** is a security breach which allows people to probe a system for users with null passwords.

A facility to allow all data exchanges to be encrypted should be present.

NAME

rfadmin – RFS domain administration

SYNOPSIS

rfadmin
rfadmin –p
rfadmin –a *hostname*
rfadmin –r *hostname*

AVAILABILITY

This program is available with the *RFS* software installation option. Refer to *Installing SunOS 4.1* for information on how to install optional software.

DESCRIPTION

rfadmin is used to add and remove hosts and their associated authentication information from a *domain*/**passwd** file on a Remote File Sharing (RFS) primary domain name server. It is also used to transfer domain name server responsibilities from one machine to another. Used with no options, **rfadmin** returns the *hostname* of the current domain name server for the local domain. For each *domain*, **/usr/nserve/auth.info/***domain*/**passwd** is created on the primary, and should be copied to all secondaries, and all hosts that want to do password verification of hosts in the *domain*.

rfadmin can only be used to modify domain files on the primary domain name server (**–a** and **–r** options). If domain name server reponsibilities are temporarily passed to a secondary domain name server, that computer can use the **–p** option to pass domain name server responsibility back to the primary. Any host can use **rfadmin** with no options to print information about the domain. The user must have **root** permissions to use the command.

Using **rfadmin** with the **–a** option, will result in an error if *hostname* is not unique in the domain.

Using **rfadmin** with the **–r** option, will send an error to the standard error if one of the following is true:

- *hostname* does not exist in the domain.
- *hostname* is defined as a domain name server.
- There are resources advertised by *hostname*.

When used with the **–p** option, **rfadmin** sends an error message to standard error, if there are no backup name servers defined for *domain*.

OPTIONS

–p Pass the domain name server responsibilities back to a primary or to a secondary name server.

–a *hostname*
Add a host to a domain that is served by this domain name server. *hostname* must be of the form *domain.nodename*. Create an entry for *hostname* in the *domain*/**passwd** file, which has the same format as **/etc/passwd**, and prompt for an initial authentication password; the password prompting process conforms with that of **passwd**(1).

–r *hostname*
Remove a host from its domain by removing it from the *domain*/**passwd** file.

FILES

/usr/nserve/auth.info/*domain*/**passwd**

SEE ALSO

passwd(1), **mount**(8), **rfstart**(8), **rfstop**(8)

NAME

rfpasswd – change RFS host password

SYNOPSIS

rfpasswd

AVAILABILITY

This program is available with the *RFS* software installation option. Refer to *Installing SunOS 4.1* for information on how to install optional software.

DESCRIPTION

rfpasswd updates the Remote File Sharing (RFS) authentication password for a host; processing of the new password follows the same criteria as **passwd**(1). The updated password is registered at the domain name server (**/usr/nserve/auth.info/***domain***/passwd**) and replaces the password stored at the local host (**/usr/nserve/loc.passwd/file**).

This command is restricted to the super-user.

Note: if you change your host password, make sure that hosts that validate your password are notified of this change. To receive the new password, hosts must obtain a copy of the *domain/passwd* file from the domain's primary name server. If this is not done, *attempts to mount remote resources may fail*.

If any of the following is true an error message will be sent to the standard error:

- The old password entered from this command does not match the existing password for this machine.
- The two new passwords entered from this command do not match.
- The new password does not satisfy the security criteria in **passwd**(1).
- The domain name server does not know about this machine.
- The command is not run with super-user privileges.

Also, RFS must be running on your host and your domain's primary name server. A new password cannot be logged if a secondary is acting as the domain name server.

FILES

/usr/nserve/auth.info/*domain***/passwd**
/usr/nserve/loc.passwd

SEE ALSO

passwd(1), **rfadmin**(8), **rfstart**(8)

NAME

rfstart – start RFS

SYNOPSIS

rfstart [–v] [–p *primary_addr*]

AVAILABILITY

This program is available with the *RFS* software installation option. Refer to *Installing SunOS 4.1* for information on how to install optional software.

DESCRIPTION

rfstart starts Remote File Sharing (RFS) and defines an authentication level for incoming requests. This command can be used only after the domain name server is set up and your computer's domain name and network specification has been defined using **dname**(8).

If the host password has not been set, **rfstart** will prompt for a password; the password prompting process must match the password entered for your machine at the primary domain name server (see **rfadmin**(8)). If you remove the **loc.passwd** file or change domains, you will also have to reenter the password.

Also, when **rfstart** is run on a domain name server, entries in the **rfmaster**(5) file are syntactically validated.

This command is restricted to the super-user.

If syntax errors are found in validating the **rfmaster**(5) file, a warning describing each error will be sent to the standard error.

An error message will be sent to the standard error if any of the following is true:

- The shared resource environment is already running.
- There is no communications network.
- The domain name server cannot be found.
- The domain name server does not recognize the machine.
- The command is run without super-user privileges.

Remote file sharing will not start if the host password in **/usr/nserve/loc.passwd** is corrupted. If you suspect this has happened, remove the file and run **rfstart** again to reenter your password.

Note: **rfstart** will *not* fail if your host password does not match the password on the domain name server. You will simply receive a warning message. However, if you try to mount a resource from the primary or any other host that validates your password, the mount will fail if your password does not match the one that host has listed for your machine.

OPTIONS

–v Specify that verification of all clients is required in response to initial incoming mount requests; any host not in the file **/usr/nserve/auth.info/***domain***/passwd** for the **domain** they belong to, will not be allowed to mount resources from your host. If the –v option is not specified, hosts named in *domain*/**passwd** will be verified, other hosts will be allowed to connect without verification.

–p *primary_addr*
Indicate the primary domain name server for your domain. *primary_addr* must be the network address of the primary name server for your domain. If the –p option is not specified, the address of the domain name server is taken from the **rfmaster** file. See **rfmaster**(5) for a description of the valid address syntax.

FILES

/usr/nserve/rfmaster
/usr/nserve/loc.passwd

SEE ALSO

rfmaster(5), adv(8), dname(8), mount(8), rfadmin(8), rfstop(8), unadv(8)

NAME

rfstop – stop the RFS environment

SYNOPSIS

rfstop

AVAILABILITY

This program is available with the *RFS* software installation option. Refer to *Installing SunOS 4.1* for information on how to install optional software.

DESCRIPTION

rfstop disconnects a host from the Remote File Sharing (RFS) environment until another **rfstart**(8) is executed.

When executed on the domain name server, the domain name server responsibility is moved to a secondary name server as designated in the **rfmaster** file.

This command is restricted to the super-user.

If any of the following is true, an error message will be sent to standard error.

- There are resources currently advertised by this host.
- Resources from this machine are still remotely mounted by other hosts.
- There are still remotely mounted resources in the local file system tree.
- **rfstart**(8) had not previously been executed.
- The command is not run with super-user privileges.

SEE ALSO

rfmaster(5), **adv**(8), **mount**(8), **rfadmin**(8), **rfstart**(8), **unadv**(8)

NAME

rfuadmin – RFS notification shell script

SYNOPSIS

rfuadmin *message remote_resource* [*seconds*]

AVAILABILITY

This program is available with the *RFS* software installation option. Refer to *Installing SunOS 4.1* for information on how to install optional software.

DESCRIPTION

The **rfuadmin** shell script is used to respond to unexpected Remote File Sharing (RFS) events picked up by the **rfudaemon**(8) process. Such events may include broken network connections and forced unmounts. This script is not intended to be run directly from the shell.

Responses to messages received by **rfudaemon** can be tailored to suit the particular system by editing this script. The following paragraphs describe the arguments passed to **rfuadmin** and its standard responses.

disconnect *remote_resource*

A link to a remote resource has been cut. **rfudaemon** executes **rfuadmin**, passing it the message **disconnect** and the name of the disconnected resource. **rfuadmin** sends this message to all terminals using **wall**(1):

remote_resource **has been disconnected from the system.**

rfuadmin executes **fuser**(8) to kill all processes using the resource, unmounts the resource, and attempts to mount the resource again.

fumount remote_resource

A remote server machine has forced an unmount of a resource a local machine has mounted. The processing is similar to processing for a disconnect.

fuwarn *remote_resource seconds*

This message notifies **rfuadmin** that a resource is about to be unmounted. **rfudaemon** sends this script the **fuwarn** message, the resource name, and the number of seconds in which the forced unmount will occur. **rfuadmin** sends this message to all terminals:

remote_resource **is being removed from the system in # seconds.**

SEE ALSO

wall(1), **fumount**(8), **fuser**(8), **mount**(8), **rfstart**(8), **rfudaemon**(8)

BUGS

The console must be on when RFS is running, otherwise **rfuadmin** hangs when it attempts to write to it, in which case recovery from disconected resources may not complete.

NAME

rfudaemon – Remote File Sharing daemon

SYNOPSIS

rfudaemon

AVAILABILITY

This program is available with the *RFS* software installation option. Refer to *Installing SunOS 4.1* for information on how to install optional software.

DESCRIPTION

The RFS daemon, **rfudaemon**, is started automatically by **rfstart**(8) and runs as a daemon process while Remote File Sharing is active. It listens for unexpected events, such as broken network connections and forced unmounts, and invokes **rfuadmin**(8) to execute the appropriate administrative procedures. Events recognized by **rfudaemon** are as follows:

disconnect

A link to a remote resource has been cut. **rfudaemon** executes **rfuadmin**, with two arguments: **disconnect** and the name of the disconnected resource.

fumount

A remote server machine has forced an unmount of a resource a local machine has mounted. **rfudaemon** executes **rfuadmin**, with two arguments: **fumount** and the name of the disconnected resource.

getumsg

A remote user-level program has sent a message to the local **rfudaemon**. Currently the only message sent is **fuwarn**, which notifies **rfuadmin** that a resource is about to be unmounted. **rfudaemon** sends **rfuadmin** the **fuwarn**, the resource name, and the number of seconds in which the forced unmount will occur.

lastumsg

The local machine wants to stop the **rfudaemon** (**rfstop**(8)). This causes **rfudaemon** to exit.

SEE ALSO

rfstart(8), **rfstop**(8), **rfuadmin**(8)

NAME

rlogind, in.rlogind – remote login server

SYNOPSIS

/usr/etc/in.rlogind *host.port*

DESCRIPTION

rlogind is the server for the **rlogin**(1C) program. The server provides a remote login facility with authentication based on privileged port numbers.

rlogind is invoked by **inetd**(8C) when a remote login connection is established, and executes the following protocol:

- The server checks the client's source port. If the port is not in the range 0-1023, the server aborts the connection. The client's address and port number are passed as arguments to **rlogind** by **inetd** in the form *host.port* with host in hex and port in decimal.
- The server checks the client's source address. If the address is associated with a host for which no corresponding entry exists in the host name data base (see **hosts**(5)), the server aborts the connection.

Once the source port and address have been checked, **rlogind** allocates a pseudo-terminal (see **pty**(4)), and manipulates file descriptors so that the slave half of the pseudo-terminal becomes the **stdin**, **stdout**, and **stderr** for a login process. The login process is an instance of the **login**(1) program, invoked with the **–r** option. The login process then proceeds with the authentication process as described in **rshd**(8C), but if automatic authentication fails, it reprompts the user to login as one finds on a standard terminal line.

The parent of the login process manipulates the master side of the pseudo-terminal, operating as an intermediary between the login process and the client instance of the **rlogin** program. In normal operation, the packet protocol described in **pty**(4) is invoked to provide ^S/^Q type facilities and propagate interrupt signals to the remote programs. The login process propagates the client terminal's baud rate and terminal type, as found in the environment variable, **TERM**; see **environ**(5V).

SEE ALSO

inetd(8C)

DIAGNOSTICS

All diagnostic messages are returned on the connection associated with the **stderr**, after which any network connections are closed. An error is indicated by a leading byte with a value of 1.

Hostname for your address unknown.
No entry in the host name database existed for the client's machine.

Try again.
A *fork* by the server failed.

/usr/bin/sh: ...
The user's login shell could not be started.

BUGS

The authentication procedure used here assumes the integrity of each client machine and the connecting medium. This is insecure, but is useful in an "open" environment.

A facility to allow all data exchanges to be encrypted should be present.

NAME

rmail – handle remote mail received via uucp

SYNOPSIS

rmail *recipient...*

DESCRIPTION

rmail interprets incoming mail received through **uucp**(1C), collapsing "From" lines in the form generated by **bin-mail (1)** (see **bin-mail**(1)) into a single line of the form *return-path*!*sender*, and passing the processed mail on to **sendmail**(8).

rmail is explicitly designed for use with **uucp**(1C) and **sendmail**(8).

SEE ALSO

bin-mail(1), **uucp**(1C), **sendmail**(8)

NAME

rm_client – remove an NFS client

SYNOPSIS

rm_client [–y] *clients*

DESCRIPTION

rm_client removes an NFS client from a server. By default, **rm_client** asks if you want to remove the client's root directory, swap file, hosts entry, and **/tftpboot** file and whether to delete the client's entry in **/etc/bootparams**. **rm_client** can be run only by the super-user on the server, while in multiuser mode, or while not in the miniroot.

OPTIONS

–y Supply "yes" answers to all questions about what to remove.

FILES

/etc/bootparams
/tftpboot/*machine_addr*
/export/root/*client*
/export/swap/*client*

SEE ALSO

add_client(8), **add_services**(8), **suninstall**(8)

Installing SunOS 4.1

DIAGNOSTICS

must be run as root (super-user).
You must be root to run **rm_client**.

NAME

rmntstat – display RFS mounted resource information

SYNOPSIS

rmntstat [**–h**] [*resource*]

AVAILABILITY

This program is available with the *RFS* software installation option. Refer to *Installing SunOS 4.1* for information on how to install optional software.

DESCRIPTION

When used with no options, **rmntstat** displays a list of all local Remote File Sharing resources that are remotely mounted, the local path name, and the corresponding clients. **rmntstat** returns the remote mount data regardless of whether a resource is currently advertised; this ensures that resources that have been unadvertised but are still remotely mounted are included in the report. When a *resource* is specified, **rmntstat** displays the remote mount information only for that resource.

This command is restricted to the super-user.

OPTIONS

–h Omit header information from the display.

EXIT STATUS

If no local resources are remotely mounted, **rmntstat** will return a successful exit status.

ERRORS

If *resource* does not physically reside on the local machine or is an invalid resource name, an error message will be sent to standard error.

SEE ALSO

mount(8), **fumount**(8), **unadv**(8)

NAME

rmt – remote magtape protocol module

SYNOPSIS

/usr/etc/rmt

DESCRIPTION

rmt is a program used by the remote dump and restore programs in manipulating a magnetic tape drive through an interprocess communication connection. **rmt** is normally started up with an **rexec**(3N) or **rcmd**(3N) call.

The **rmt** program accepts requests specific to the manipulation of magnetic tapes, performs the commands, then responds with a status indication. All responses are in ASCII and in one of two forms. Successful commands have responses of

A*number*\n

where *number* is an ASCII representation of a decimal number. Unsuccessful commands are responded to with

E*error-number*\n*error-message*\n

where *error-number* is one of the possible error numbers described in **intro**(2) and *error-message* is the corresponding error string as printed from a call to **perror**(3). The protocol is comprised of the following commands:

S Return the status of the open device, as obtained with a **MTIOCGET ioctl** call. If the operation was successful, an "ack" is sent with the size of the status buffer, then the status buffer is sent (in binary).

C*device* Close the currently open device. The *device* specified is ignored.

I*operation*\n*count*\n
Perform a **MTIOCOP ioctl**(2) command using the specified parameters. The parameters are interpreted as the ASCII representations of the decimal values to place in the *mt_op* and *mt_count* fields of the structure used in the **ioctl** call. The return value is the *count* parameter when the operation is successful.

L*whence*\n*offset*\n
Perform an **lseek**(2V) operation using the specified parameters. The response value is that returned from the **lseek** call.

O*device*\n*mode*\n
Open the specified *device* using the indicated *mode*. *device* is a full pathname and *mode* is an ASCII representation of a decimal number suitable for passing to **open**(2V). If a device had already been opened, it is closed before a new open is performed.

R*count* Read *count* bytes of data from the open device. **rmt** performs the requested **read**(2V) and responds with A*count-read*\n if the read was successful; otherwise an error in the standard format is returned. If the read was successful, the data read is then sent.

W*count* Write data onto the open device. **rmt** reads *count* bytes from the connection, aborting if a premature EOF is encountered. The response value is that returned from the **write**(2V) call.

Any other command causes **rmt** to exit.

DIAGNOSTICS

All responses are of the form described above.

SEE ALSO

intro(2), **ioctl**(2), **lseek**(2V), **open**(2V), **read**(2V), **write**(2V), **perror**(3), **rcmd**(3N), **rexec**(3N), **mtio**(4), **dump**(8), **restore**(8)

BUGS

People tempted to use this for a remote file access protocol are discouraged.

NAME

route – manually manipulate the routing tables

SYNOPSIS

/usr/etc/route [**–fn**] **add** | **delete** [**host** | **net**] *destination* [*gateway* [*metric*]]

AVAILABILITY

This program is available with the *Networking* software installation option. Refer to *Installing SunOS 4.1* for information on how to install optional software.

DESCRIPTION

route manually manipulates the network routing tables normally maintained by the system routing daemon, **routed**(8C), or through default routes and redirect messages from routers. **route** allows the superuser to operate directly on the routing table for the specific host or network indicated by *destination*. The *gateway* argument, if present, indicates the network gateway to which packets should be addressed. The *metric* argument indicates the number of "hops" to the *destination*. The *metric* is required for *add* commands; it must be zero if the destination is on a directly-attached network, and nonzero if the route utilizes one or more gateways.

The **add** command instructs **route** to add a route to *destination*. **delete** deletes a route.

Routes to a particular host must be distinguished from those to a network. The optional keywords **net** and **host** force the destination to be interpreted as a network or a host, respectively. Otherwise, if the destination has a "local address part" of **INADDR_ANY**, then the route is assumed to be to a network; otherwise, it is presumed to be a route to a host. If the route is to a destination connected by a gateway, the *metric* parameter should be greater than 0. If adding a route with metric 0, the gateway given is the address of this host on the common network, indicating the interface to be used directly for transmission. All symbolic names specified for a *destination* or *gateway* are looked up in the hosts database using **gethostbyname**( ) (see **gethostent**(3N)). If this lookup fails, then the name is looked up in the networks database using **getnetbyname**( ) (see **getnetent**(3N)). "default" is also a valid destination, which is used for all routes if there is no specific host or network route.

OPTIONS

–f Flush the routing tables of all gateway entries. If this is used in conjunction with one of the commands described above, **route** flushes the gateways before performing the command.

–n Prevents attempts to print host and network names symbolically when reporting actions. This is useful, for example, when all name servers are down on your local net, so you need a route before you can contact the name server.

FILES

/etc/hosts
/etc/networks

SEE ALSO

ioctl(2), **gethostent**(3N), **getnetent**(3N), **routing**(4N), **routed**(8C)

DIAGNOSTICS

add [**host** | **net**] *destination*:*gateway*
The specified route is being added to the tables. The values printed are from the routing table entry supplied in the **ioctl**(2) call.

delete [**host** | **net**] *destination*:*gateway*
The specified route is being deleted.

*destination***done**
When the –f flag is specified, each routing table entry deleted is indicated with a message of this form.

Network is unreachable

An attempt to add a route failed because the gateway listed was not on a directly-connected network. Give the next-hop gateway instead.

not in table

A delete operation was attempted for an entry that is not in the table.

routing table overflow

An add operation was attempted, but the system was unable to allocate memory to create the new entry.

NAME

routed, in.routed – network routing daemon

SYNOPSIS

/usr/etc/in.routed [**–qstv**] [*logfile*]

DESCRIPTION

routed is invoked at boot time to manage the network routing tables. The routing daemon uses a variant of the Xerox NS Routing Information Protocol in maintaining up to date kernel routing table entries.

In normal operation **routed** listens on **udp**(4P) socket 520 (decimal) for routing information packets. If the host is an internetwork router, it periodically supplies copies of its routing tables to any directly connected hosts and networks.

When **routed** is started, it uses the **SIOCGIFCONF ioctl**( ) (see **ioctl**(2)) to find those directly connected interfaces configured into the system and marked "up" (the software loopback interface is ignored). If multiple interfaces are present, it is assumed the host will forward packets between networks. **routed** then transmits a *request* packet on each interface (using a broadcast packet if the interface supports it) and enters a loop, listening for *request* and *response* packets from other hosts.

When a *request* packet is received, **routed** formulates a reply based on the information maintained in its internal tables. The *response* packet generated contains a list of known routes, each marked with a "hop count" metric (a count of 16, or greater, is considered "infinite"). The metric associated with each route returned provides a metric *relative to the sender*.

request packets received by **routed** are used to update the routing tables if one of the following conditions is satisfied:

- No routing table entry exists for the destination network or host, and the metric indicates the destination is "reachable" (that is, the hop count is not infinite).
- The source host of the packet is the same as the router in the existing routing table entry. That is, updated information is being received from the very internetwork router through which packets for the destination are being routed.
- The existing entry in the routing table has not been updated for some time (defined to be 90 seconds) and the route is at least as cost effective as the current route.
- The new route describes a shorter route to the destination than the one currently stored in the routing tables; the metric of the new route is compared against the one stored in the table to decide this.

When an update is applied, **routed** records the change in its internal tables and generates a *response* packet to all directly connected hosts and networks. **routed** waits a short period of time (no more than 30 seconds) before modifying the kernel's routing tables to allow possible unstable situations to settle.

In addition to processing incoming packets, **routed** also periodically checks the routing table entries. If an entry has not been updated for 3 minutes, the entry's metric is set to infinity and marked for deletion. Deletions are delayed an additional 60 seconds to insure the invalidation is propagated throughout the internet.

Hosts acting as internetwork routers gratuitously supply their routing tables every 30 seconds to all directly connected hosts and networks.

In addition to the facilities described above, **routed** supports the notion of "distant" *passive* and *active* gateways. When **routed** is started up, it reads the file **/etc/gateways** to find gateways which may not be identified using the **SIOGIFCONF ioctl**( ). Gateways specified in this manner should be marked passive if they are not expected to exchange routing information, while gateways marked active should be willing to exchange routing information (that is, they should have a **routed** process running on the machine). Passive gateways are maintained in the routing tables forever and information regarding their existence is included in any routing information transmitted. Active gateways are treated equally to network interfaces. Routing information is distributed to the gateway and if no routing information is received for a period of the time, the associated route is deleted.

The **/etc/gateways** is comprised of a series of lines, each in the following format:

< net | host > *filename1* **gateway** *filename2* **metric** *value* **< passive | active >**

The **net** or **host** keyword indicates if the route is to a network or specific host.

filename1 is the name of the destination network or host. This may be a symbolic name located in **/etc/networks** or **/etc/hosts**, or an Internet address specified in "dot" notation; see **inet**(3N).

filename2 is the name or address of the gateway to which messages should be forwarded.

value is a metric indicating the hop count to the destination host or network.

The keyword **passive** or **active** indicates if the gateway should be treated as passive or active (as described above).

OPTIONS

–s Force **routed** to supply routing information whether it is acting as an internetwork router or not.

–q Opposite of the **–s** option.

–t All packets sent or received are printed on the standard output. In addition, **routed** will not divorce itself from the controlling terminal so that interrupts from the keyboard will kill the process.

–v Allow a logfile to be created showing the changes made to the routing tables with a timestamp.

logfile Specify a file in which **routed** records any changes to the routing tables and a history of recent messages sent and received which are related to the changed route.

FILES

/etc/gateways	for distant gateways
/etc/networks	
/etc/hosts	

SEE ALSO

ioctl(2), **inet**(3N), **udp**(4P)

BUGS

The kernel's routing tables may not correspond to those of **routed** for short periods of time while processes utilizing existing routes exit; the only remedy for this is to place the routing process in the kernel.

routed should listen to intelligent interfaces, such as an IMP, and to error protocols, such as ICMP, to gather more information.

NAME

rpcinfo – report RPC information

SYNOPSIS

rpcinfo **–p** [*host*]

rpcinfo [**–n** *portnum*] **–u** *host program* [*version*]

rpcinfo [**–n** *portnum*] **–t** *host program* [*version*]

rpcinfo **–b** *program version*

rpcinfo **–d** *program version*

AVAILABILITY

This program is available with the *Networking* software installation option. Refer to *Installing SunOS 4.1* for information on how to install optional software.

DESCRIPTION

rpcinfo makes an RPC call to an RPC server and reports what it finds.

OPTIONS

–p Probe the portmapper on *host*, and print a list of all registered RPC programs. If *host* is not specified, it defaults to the value returned by **hostname**(1).

–u Make an RPC call to procedure 0 of *program* on the specified *host* using UDP, and report whether a response was received.

–t Make an RPC call to procedure 0 of *program* on the specified *host* using TCP, and report whether a response was received.

–n Use *portnum* as the port number for the *–t* and *–u* options instead of the port number given by the portmapper.

–b Make an RPC broadcast to procedure 0 of the specified *program* and *version* using UDP and report all hosts that respond.

–d Delete registration for the RPC service of the specified *program* and *version*. This option can be exercised only by the super-user.

The *program* argument can be either a name or a number.

If a *version* is specified, **rpcinfo** attempts to call that version of the specified *program*. Otherwise, **rpcinfo** attempts to find all the registered version numbers for the specified *program* by calling version 0 (which is presumed not to exist; if it does exist, **rpcinfo** attempts to obtain this information by calling an extremely high version number instead) and attempts to call each registered version. Note: the version number is required for **–b** and **–d** options.

EXAMPLES

To show all of the RPC services registered on the local machine use:

example% rpcinfo –p

To show all of the RPC services registered on the machine named **klaxon** use:

example% rpcinfo –p klaxon

To show all machines on the local net that are running the Network Interface Service (NIS) use:

example% rpcinfo –b ypserv 'version' | uniq

where 'version' is the current NIS version obtained from the results of the **–p** switch above.

To delete the registration for version 1 of the **walld** service use:

example% rpcinfo –d walld 1

SEE ALSO

rpc(5), **portmap**(8C)

RPC Programming Guide in *Network Programming*

BUGS

In releases prior to the SunOS 3.0 release, the Network File System (NFS) did not register itself with the portmapper; **rpcinfo** cannot be used to make RPC calls to the NFS server on hosts running such releases.

NOTES

The Network Information Service (NIS) was formerly known as Sun Yellow Pages (YP). The functionality of the two remains the same; only the name has changed.

NAME

rquotad, rpc.rquotad – remote quota server

SYNOPSIS

/usr/etc/rpc.rquotad

AVAILABILITY

This program is available with the *Networking* software installation option. Refer to *Installing SunOS 4.1* for information on how to install optional software.

DESCRIPTION

rquotad is an **rpc**(3N) server which returns quotas for a user of a local file system which is mounted by a remote machine over the NFS. The results are used by **quota**(1) to display user quotas for remote file systems. The **rquotad** daemon is normally invoked by **inetd**(8C).

FILES

quotas	quota file at the file system root

SEE ALSO

quota(1), **rpc**(3N), **nfs**(4P), **services**(5) **inetd**(8C)

NAME

rshd, in.rshd – remote shell server

SYNOPSIS

/usr/etc/in.rshd *host.port*

DESCRIPTION

rshd is the server for the **rcmd**(3N) routine and, consequently, for the **rsh**(1C) program. The server provides remote execution facilities with authentication based on privileged port numbers.

rshd is invoked by **inetd**(8C) each time a shell service is requested, and executes the following protocol:

- The server checks the client's source port. If the port is not in the range 512-1023, the server aborts the connection. The clients host address (in hex) and port number (in decimal) are the argument passed to **rshd.**
- The server reads characters from the socket up to a null (**\0**) byte. The resultant string is interpreted as an ASCII number, base 10.
- If the number received in step 1 is non-zero, it is interpreted as the port number of a secondary stream to be used for the **stderr**. A second connection is then created to the specified port on the client's machine. The source port of this second connection is also in the range 512-1023.
- The server checks the client's source address. If the address is associated with a host for which no corresponding entry exists in the host name data base (see **hosts**(5)), the server aborts the connection.
- A null terminated user name of at most 16 characters is retrieved on the initial socket. This user name is interpreted as a user identity to use on the **server**'s machine.
- A null terminated user name of at most 16 characters is retrieved on the initial socket. This user name is interpreted as the user identity on the **client**'s machine.
- A null terminated command to be passed to a shell is retrieved on the initial socket. The length of the command is limited by the upper bound on the size of the system's argument list.
- **rshd** then validates the user according to the following steps. The remote user name is looked up in the password file and a **chdir** is performed to the user's home directory. If the lookup or fails, the connection is terminated. If the **chdir** fails, it does a **chdir** to / **(root).** If the user is not the super-user, (user ID 0), the file **/etc/hosts.equiv** is consulted for a list of hosts considered "equivalent". If the client's host name is present in this file, the authentication is considered successful. If the lookup fails, or the user is the super-user, then the file **.rhosts** in the home directory of the remote user is checked for the machine name and identity of the user on the client's machine. If this lookup fails, the connection is terminated.
- A null byte is returned on the connection associated with the **stderr** and the command line is passed to the normal login shell of the user. The shell inherits the network connections established by **rshd.**

FILES

/etc/hosts.equiv

SEE ALSO

rsh(1C), **rcmd**(3N), **syslogd**(8)

BUGS

The authentication procedure used here assumes the integrity of each client machine and the connecting medium. This is insecure, but is useful in an "open" environment.

A facility to allow all data exchanges to be encrypted should be present.

DIAGNOSTICS

The following diagnostic messages are returned on the connection associated with the **stderr**, after which any network connections are closed. An error is indicated by a leading byte with a value of 1 (0 is returned in step 9 above upon successful completion of all the steps prior to the command execution).

locuser too long
The name of the user on the client's machine is longer than 16 characters.

remuser too long
The name of the user on the remote machine is longer than 16 characters.

command too long
The command line passed exceeds the size of the argument list (as configured into the system).

Hostname for your address unknown.
No entry in the host name database existed for the client's machine.

Login incorrect.
No password file entry for the user name existed.

Permission denied.
The authentication procedure described above failed.

Can't make pipe.
The pipe needed for the **stderr**, was not created.

Try again.
A *fork* by the server failed.

/usr/bin/sh: ...
The user's login shell could not be started.

In addition, daemon's status messages and internal diagnostics are logged to the appropriate system log using the **syslogd**(8) facility.

NAME

rstatd, rpc.rstatd – kernel statistics server

SYNOPSIS

/usr/etc/rpc.rstatd

DESCRIPTION

rstatd is a server which returns performance statistics obtained from the kernel. These statistics are graphically displayed by **perfmeter**(1). The **rstatd** daemon is normally invoked by **inetd**(8C).

Systems with disk drivers to be monitored by this daemon must be configured so as to report disk (**_dk_xfer**) statistics.

SEE ALSO

perfmeter(1), **services**(5), **inetd**(8C)

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 using **cron**(8). **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 **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 the **wtmp** file, correcting date changes if necessary.
CONNECT1	Produce connect session records in **ctmp.h** format.
CONNECT2	Convert **ctmp.h** records into **tacct.h** format.
PROCESS	Convert process accounting records into **tacct.h** format.
MERGE	Merge the connect and process accounting records.
FEES	Convert output of **chargefee** into **tacct.h** 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	Cleanup temporary files and exit.

To restart **runacct** after a failure, first check the **active** file for diagnostics, then fix up any corrupted data files, such as **pacct** or **wtmp**. The **lock** files and **lastdate** file 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. 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 &

FILES

/etc/wtmp
/var/adm/pacct*
/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
/var/adm/acct/nite/ptacct*.*mmdd*

SEE ALSO

acctcom(1), **mail**(1), **acct**(2V), **acct**(5), **utmp**(5V), **acct**(8), **acctcms**(8), **acctcon**(8), **acctmerg**(8), **acctprc**(8), **acctsh**(8), **cron**(8), **fwtmp**(8)

BUGS

Normally it is not a good idea to restart **runacct** in the **SETUP** *state*. Run **SETUP** manually and restart using:

runacct *mmdd* **WTMPFIX**

If **runacct** failed in the **PROCESS** *state*, remove the last **ptacct** file because it will not be complete.

NAME

rusage – print resource usage for a command

SYNOPSIS

rusage *command*

DESCRIPTION

The **rusage** command is similar to **time**(1V). It runs the given *command*, which must be specified; that is, *command* is not optional as it is in the C shell's timing facility. When the command is complete, **rusage** displays the real (wall clock), the system CPU, and the user CPU times which elapsed during execution of the command, plus other fields in the **rusage** structure, all on one long line. Times are reported in seconds and hundredths of a second.

EXAMPLE

The example below shows the format of **rusage** output.

```
example% rusage wc /usr/man/man1/csh (1)
3045  13423  78071 /usr/man/man1/csh (1)
2.26 real 0.80 user 0.36 sys 11 pf 38 pr 0 sw 11 rb 0 wb 16 vcx 37 icx 24 mx 0 ix 1230 id 9 is
example%
```

Each of the fields identified corresponds to an element of the **rusage** structure, as described in **getrusage**(2), as follows:

real		**elapsed real time**
user	**ru_utime**	**user time used**
sys	**ru_stime**	**system time used**
pf	**ru_majflt**	**page faults requiring physical I/O**
pr	**ru_minflt**	**page faults not requiring physical I/O**
sw	**ru_nswap**	**swaps**
rb	**ru_inblock**	**block input operations**
wb	**ru_oublock**	**block output operations**
vcx	**ru_nvcsw**	**voluntary context switches**
icx	**ru_nivcsw**	**involuntary context switches**
mx	**ru_maxrss**	**maximum resident set size**
ix	**ru_ixrss**	**currently 0**
id	**ru_idrss**	**integral resident set size**
is	**ru_isrss**	**currently 0**

SEE ALSO

csh(1), **time**(1V), **getrusage**(2)

BUGS

When the command being timed is interrupted, the timing values displayed may be inaccurate.

NAME

rusersd, rpc.rusersd – network username server

SYNOPSIS

/usr/etc/rpc.rusersd

AVAILABILITY

This program is available with the *Networking* software installation option. Refer to *Installing SunOS 4.1* for information on how to install optional software.

DESCRIPTION

rusersd is a server that returns a list of users on the network. The **rusersd** daemon is normally invoked by **inetd**(8C).

SEE ALSO

perfmeter(1), **rusers**(1C), **services**(5) **inetd**(8C)

Installing SunOS 4.1

NAME

rwalld, rpc.rwalld – network rwall server

SYNOPSIS

/usr/etc/rpc.rwalld

AVAILABILITY

This program is available with the *Networking* software installation option. Refer to *Installing SunOS 4.1* for information on how to install optional software.

DESCRIPTION

rwalld is a server that handles **rwall**(1C) and **shutdown**(2) requests. It is implemented by calling **wall**(1) to all the appropriate network machines. The **rwalld** daemon is normally invoked by **inetd**(8C).

SEE ALSO

rwall(1C), **wall**(1), **shutdown**(2) **services**(5), **inetd**(8C)

NAME

rwhod, in.rwhod – system status server

SYNOPSIS

/usr/etc/in.rwhod

AVAILABILITY

Due to its potential impact on network performance, this service is commented out of the **/etc/rc** system initialization script. It is provided only for 4.3 BSD compatibility.

This program is available with the *Networking* software installation option. Refer to *Installing SunOS 4.1* for information on how to install optional software.

DESCRIPTION

rwhod is the server which maintains the database used by the **rwho**(1C) and **ruptime**(1C) programs. Its operation is predicated on the ability to *broadcast* messages on a network.

rwhod operates as both a producer and consumer of status information. As a producer of information it periodically queries the state of the system and constructs status messages which are broadcast on a network. As a consumer of information, it listens for other **rwhod** servers' status messages, validating them, then recording them in a collection of files located in the directory **/var/spool/rwho**.

The **rwho** server transmits and receives messages at the port indicated in the "rwho" service specification, see **services**(5). The messages sent and received, are of the form:

```
struct  outmp {
        char    out_line[8];    /* tty name */
        char    out_name[8];    /* user id */
        long    out_time;       /* time on */
};

struct  whod {
        char    wd_vers;
        char    wd_type;
        char    wd_fill[2];
        int     wd_sendtime;
        int     wd_recvtime;
        char    wd_hostname[32];
        int     wd_loadav[3];
        int     wd_boottime;
                struct  whoent {
                struct  outmp we_utmp;
                int     we_idle;
        } wd_we[1024 / sizeof (struct whoent)];
};
```

All fields are converted to network byte order prior to transmission. The load averages are as calculated by the w(1) program, and represent load averages over the 5, 10, and 15 minute intervals prior to a server's transmission. The host name included is that returned by the **gethostname**(2) system call. The array at the end of the message contains information about the users logged in to the sending machine. This information includes the contents of the **utmp**(5V) entry for each non-idle terminal line and a value indicating the time since a character was last received on the terminal line.

Messages received by the **rwho** server are discarded unless they originated at a **rwho** server's port. In addition, if the host's name, as specified in the message, contains any unprintable ASCII characters, the message is discarded. Valid messages received by **rwhod** are placed in files named **whod.***hostname* in the directory **/var/spool/rwho**. These files contain only the most recent message, in the format described above.

Status messages are generated approximately once every 60 seconds. **rwhod** performs an **nlist** (3V) on **/vmunix** every 10 minutes to guard against the possibility that this file is not the system image currently operating.

FILES

/etc/rc
/var/spool/rwho

SEE ALSO

rwho(1C), **ruptime**(1C), w(1), **gethostname**(2), **nlist**(3V), **utmp**(5V), **syslogd**(8)

DIAGNOSTICS

Status and diagnostic messages are logged to the appropriate system log using the **syslogd**(8) facility.

BUGS

This service takes up progressively more network bandwidth as the number of hosts on the local net increases. For large networks, the cost becomes prohibitive. RPC-based services such as **rup**(1C) and **rusers**(1C) provide a similar function with greater efficiency.

rwhod should relay status information between networks. People often interpret the server dying as a machine going down.

NAME

sa, accton – system accounting

SYNOPSIS

/usr/etc/sa [**–abcdDfijkKlmnrstu**] [**–v**[*n*]] [**–S** *savacctfile*] [**–U** *usracctfile*] [*filename*]

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

DESCRIPTION

With an argument naming an existing *filename*, **accton** causes system accounting information for every process executed to be placed at the end of the file. If no argument is given, accounting is turned off.

sa reports on, cleans up, and generally maintains accounting files.

sa is able to condense the information in **/var/adm/pacct** into a summary file **/var/adm/savacct** which contains a count of the number of times each command was called and the time resources consumed. This condensation is desirable because on a large system **/var/adm/pacct** can grow by 500K bytes per day. The summary file is normally read before the accounting file, so the reports include all available information.

If a file name is given as the last argument, that file will be treated as the accounting file; **/var/adm/pacct** is the default.

Output fields are labeled: **cpu** for the sum of user+system time (in minutes), **re** for real time (also in minutes), **k** for CPU-time averaged core usage (in 1k units), **avio** for average number of I/O operations per execution. With options fields labeled **tio** for total I/O operations, **k*sec** for CPU storage integral (kilo-core seconds), **u** and **s** for user and system CPU time alone (both in minutes) will sometimes appear.

sa also breaks out accounting statistics by user. This information is kept in the file **/var/adm/usracct**.

OPTIONS

–a Print all command names, even those containing unprintable characters and those used only once. By default, those are placed under the name '***other.'

–b Sort output by sum of user and system time divided by number of calls. Default sort is by sum of user and system times.

–c Besides total user, system, and real time for each command print percentage of total time over all commands.

–d Sort by average number of disk I/O operations.

–D Print and sort by total number of disk I/O operations.

–f Force no interactive threshold compression with **–v** flag.

–i Do not read in summary file.

–j Instead of total minutes time for each category, give seconds per call.

–k Sort by CPU-time average memory usage.

–K Print and sort by CPU-storage integral.

–l Separate system and user time; normally they are combined.

–m Print number of processes and number of CPU minutes for each user.

–n Sort by number of calls.

–r Reverse order of sort.

–s Merge accounting file into summary file **/var/adm/savacct** when done.

–t For each command report ratio of real time to the sum of user and system times.

–u Superseding all other flags, print for each record in the accounting file the user ID and command name.

–v Followed by a number *n*, types the name of each command used *n* times or fewer. If *n* is not specified, it defaults to 1. Await a reply from the terminal; if it begins with y, add the command to the category '**junk**.' This is used to strip out garbage.

–S The following filename is used as the command summary file instead of **/var/adm/savacct**.

–U The following filename is used instead of **/var/adm/usracct** to accumulate the per-user statistics printed by the **–m** option.

FILES

/var/adm/pacct	raw accounting
/var/adm/savacct	summary by command
/var/adm/usracct	summary by user ID

SEE ALSO

acct(2V), **acct**(5), **ac**(8)

BUGS

sa's execution time increases linearly with the magnitude of the largest positive user ID in **/etc/passwd**.

NAME

savecore – save a core dump of the operating system

SYNOPSIS

/usr/etc/savecore [–v] *directory* [*system-name*]

DESCRIPTION

savecore saves a core dump of the kernel (assuming that one was made) and writes a reboot message in the shutdown log. It is meant to be called near the end of the **/etc/rc.local** file after the system boots. However, it is not normally run by default. You must edit that file to enable it.

savecore checks the core dump to be certain it corresponds with the version of the operating system currently running. If it does, **savecore** saves the core image in the file *directory*/**vmcore.***n* and the kernel's namelist in *directory*/**vmunix.***n*. The trailing **.***n* in the pathnames is replaced by a number which grows every time **savecore** is run in that directory.

Before **savecore** writes out a core image, it reads a number from the file *directory*/**minfree**. This is the minimum number of kilobytes that must remain free on the filesystem containing *directory*. If there is less free space on the filesystem containing *directory* than the number of kilobytes specified in **minfree**, the core dump is not saved. If the **minfree** file does not exist, **savecore** always writes out the core file (assuming that a core dump was taken).

savecore also logs a reboot message using facility LOG_AUTH (see **syslog**(3)). If the system crashed as a result of a panic, **savecore** logs the panic string too.

If the core dump was from a system other than **/vmunix**, the name of that system must be supplied as *system-name*.

OPTIONS

–v	Verbose. Enable verbose error messages from **savecore**.

FILES

directory/**vmcore.***n*	
directory/**vmunix.***n*	
directory/**minfree**	
/vmunix	the kernel
/etc/rc.local	

SEE ALSO

syslog(3), **panic**(8S), **sa**(8)

BUGS

savecore can be fooled into thinking a core dump is the wrong size.

You must run **savecore** very soon after booting — before the swap space containing the crash dump is overwritten by programs currently running.

NAME

sendmail – send mail over the internet

SYNOPSIS

/usr/lib/sendmail [**–ba**] [**–bd**] [**–bi**] [**–bm**] [**–bp**] [**–bs**] [**–bt**] [**–bv**] [**–bz**]
[**–C***file*] [**–d***X*] [**–F***fullname*] [**–f***name*] [**–h***N*] [**–n**] [**–o***x value*] [**–q**[*time*]]
[**–r***name*] [**–R***string*] [**–t**] [**–v**] [*address* ...]

DESCRIPTION

sendmail sends a message to one or more people, routing the message over whatever networks are necessary. **sendmail** does internetwork forwarding as necessary to deliver the message to the correct place.

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 flags, **sendmail** reads its standard input up to an EOF, or a line with a single dot and sends a copy of the letter found there to all of the addresses listed. It determines the network to use based on the syntax and contents of the addresses.

Local addresses are looked up in the local **aliases**(5) file, or by using the Network Interface Service (NIS), and aliased appropriately. In addition, if there is a **.forward** file in a recipient's home directory, **sendmail** forwards a copy of each message to the list of recipients that file contains. Aliasing can be prevented by preceding the address with a backslash. Normally the sender is not included in alias expansions, for example, if 'john' sends to 'group', and 'group' includes 'john' in the expansion, then the letter will not be delivered to 'john'.

sendmail will also route mail directly to other known hosts in a local network. The list of hosts to which mail is directly sent is maintained in the file **/usr/lib/mailhosts**.

OPTIONS

–ba Go into ARPANET mode. All input lines must end with a LINEFEED, 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, waiting for incoming SMTP connections.

–bi Initialize the alias database.

–bm Deliver mail in the usual way (default).

–bp Print a summary of the mail queue.

–bs Use the SMTP protocol as described in RFC 821. This flag implies all the operations of the **–ba** flag 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 — do not try to collect or deliver a message. Verify mode is normally used for validating users or mailing lists.

–bz Create the configuration freeze file.

–C*file* Use alternate configuration file.

–d*X* Set debugging value to *X*.

–F*fullname* Set the full name of the sender.

–f*name* Sets the name of the "from" person (that is, the sender of the mail). **–f** can only be used by "trusted" users (who are listed in the config file).

–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.

−M*id* Attempt to deliver the queued message with message-id **id**.

−n Do not do aliasing.

−o*x value* Set option *x* to the specified *value*. Options are described below.

−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.

−r*name* An alternate and obsolete form of the **−f** flag.

−R*string* Go through the queue of pending mail and attempt to deliver any message with a recipient containing the specified string. This is useful for clearing out mail directed to a machine which has been down for awhile.

−t Read message for recipients. "To:", "Cc:", and "Bcc:" lines will be scanned for people to send to. The "Bcc:" line will be deleted before transmission. Any addresses in the argument list will be suppressed.

−v Go into verbose mode. Alias expansions will be announced, etc.

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** flag or in the configuration file. These are described in detail in the *Installation and Operation Guide*. The options are:

A*file* Use alternate alias file.

c On mailers that are considered "expensive" to connect to, do not initiate immediate connection. This requires queueing.

d*x* Set the delivery mode to **x**. Delivery modes are **i** for interactive (synchronous) delivery, **b** for background (asynchronous) delivery, and **q** for queue only — that is, actual delivery is done the next time the queue is run.

D Run **newaliases**(8) to automatically rebuild the alias database, if necessary.

e*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.

F*mode* The mode to use when creating temporary files.

f Save UNIX-system-style "From" lines at the front of messages.

g*N* The default group ID to use when calling mailers.

H*file* The SMTP help file.

i Do not take dots on a line by themselves as a message terminator.

L*n* The log level.

m Send to "me" (the sender) also if I am in an alias expansion.

o If set, this message may have old style headers. If not set, this message is guaranteed to have new style headers (that is, commas instead of spaces between addresses). If set, an adaptive algorithm is used that will correctly determine the header format in most cases.

Q*queuedir*
Select the directory in which to queue messages.

r*timeout*
: The timeout on reads; if none is set, **sendmail** will wait forever for a mailer.

S*file* Save statistics in the named file.

s Always instantiate the queue file, even under circumstances where it is not strictly necessary.

T*time* Set the timeout on messages in the queue to the specified time. After sitting in the queue for this amount of time, they will be returned to the sender. The default is three days.

t*stz,dtz* Set the name of the time zone.

u*N* Set the default user id for mailers.

If the first character of the user name is a vertical bar, the rest of the user name is used as the name of a program to pipe the mail to. It may be necessary to quote the name of the user to keep **sendmail** from suppressing the blanks from between arguments.

sendmail returns an exit status describing what it did. The codes are defined in **sysexits.h**

EX_OK	Successful completion on all addresses.
EX_NOUSER	User name not recognized.
EX_UNAVAILABLE	Catchall meaning necessary resources were not available.
EX_SYNTAX	Syntax error in address.
EX_SOFTWARE	Internal software error, including bad arguments.
EX_OSERR	Temporary operating system error, such as "cannot fork".
EX_NOHOST	Host name not recognized.
EX_TEMPFAIL	Message could not be sent immediately, but was queued.

If invoked as **newaliases**, **sendmail** rebuilds the alias database. If invoked as **mailq**, **sendmail** prints the contents of the mail queue.

FILES

Except for **/etc/sendmail.cf**, these pathnames are all specified in **/etc/sendmail.cf**. Thus, these values are only approximations.

/etc/aliases	raw data for alias names
/etc/aliases.pag	data base of alias names
/etc/aliases.dir	
/usr/lib/mailhosts	list of hosts to which mail can be sent directly
/etc/sendmail.cf	configuration file
/etc/sendmail.fc	frozen configuration
/etc/sendmail.hf	help file
/etc/sendmail.st	collected statistics
/usr/bin/uux	to deliver uucp mail
/usr/bin/mail	to deliver local mail
/var/spool/mqueue/*	temp files and queued mail
~/.forward	list of recipients for forwarding messages

SEE ALSO

biff(1), **bin-mail**(1), **mail**(1), **aliases**(5) **newaliases**(8)

System and Network Administration

Su, Zaw-Sing, and Jon Postel, *The Domain Naming Convention for Internet User Applications*, RFC 819, Network Information Center, SRI International, Menlo Park, Calif., August 1982.

Postel, Jon, *Simple Mail Transfer Protocol*, RFC 821, Network Information Center, SRI International, Menlo Park, Calif., August 1982.

Crocker, Dave, *Standard for the Format of ARPA-Internet Text Messages*, RFC 822, Network Information Center, SRI International, Menlo Park, Calif., August 1982.

NOTES

The Network Information Service (NIS) was formerly known as Sun Yellow Pages (YP). The functionality of the two remains the same; only the name has changed.

NAME

set4, unset4, check4 – set, unset, and check the 4 megabyte process virtual address space limit flag in a Sun386i module

SYNOPSIS

set4 [**–d** *working_directory*] [– l *filename*] ...

unset4 *filename* ...

check4 *filename* ...

AVAILABILITY

Available only on Sun 386i systems running a SunOS 4.0.*x* release or earlier. Not a SunOS 4.1 release feature.

DESCRIPTION

set4 sets the 4 megabyte process memory flag in each *filename* program image, limiting the virtual address space for each program to 4 megabytes. If a '–' is used, **set4** reads the standard input for a list of files to set the 4 megabyte limit on. Lines in the standard input whose first character is '#' are ignored, so files may include comments.

unset4 clears the 4 megabyte process memory flag in the program image, so the process virtual address space is not limited to 4 megabytes.

check4 reports programs that do not have the 4 megabyte limit set, and does not report programs with the limit set.

OPTIONS

–d *working_directory*

This specifies a directory prefix for file names that **set4** processes.

EXAMPLES

Suppose that the file **small_progs** contains the following:

```
# These files should have their virtual address spaces limited to 4 MB:
/bin/date
/bin/true
```

Then the following command will run **set4** on **/build/bin/false**, **/build/bin/date**, **/build/bin/true**, and **/build/bin/cat**.

```
example% set4 –d /build /bin/false –
/bin/cat < small_progs
example%
```

In this example, **unset4** clears the 4 megabyte limit flag in **date**, and **clri**.

```
example% unset4 /bin/date /etc/clri
example%
```

In the last example, **check4** shows that **date** and **clri** are 4 megabyte processes, but **basename** is not.

```
example% check4 /bin/date /etc/clri /usr/bin/basename
basename is not a 4MB process
example%
```

SEE ALSO

execve(2V) **execl**(3V)

BUGS

There is a problem in the way that processes that have the 4 megabyte limit set exec( ) processes that do not have the limit set. (See **execve**(2V) and **execl**(3V) for descriptions of exec( ) processing.) For a short time during the exec( ), a child has the parent's data and stack limits. During this time, the program is checked

to see if it will fit into memory. If the parent had the 4 megabyte limit set, the test fails, because the child program is running with the parent's 4 megabyte limit. This only affects programs which have more than 4 megabytes of global or static data compiled into the program. It does not affect programs which use **malloc**(3V) to obtain memory.

For example, **csh**(1) and **sh**(1) may be 4 megabyte processes. If they are, and if you try to run a program with more than 4 megabytes of global and static data, the shell cannot successfully **exec**( ). To fix this problem, become root on your machine and enter the following commands:

```
example% /etc/mount -o remount,rw /usr
/usr/etc/unset4 /bin/csh /bin/sh
example%
```

Then log out and back in again to run the modified shell. This makes **csh** and **sh** "normal" processes.

NAME

setsid – set process to session leader

SYNOPSIS

setsid [**–b**] *command* [*arguments*]

DESCRIPTION

setsid executes *command* after altering the execution environment such that the next non-controlling terminal opened will be assigned as *command*'s controlling terminal.

OPTIONS

–b Alteration to the execution environment persists across calls to **fork**(2V).

The **–b** option puts the process into a state that is supported in SunOS Release 4.1 solely as a migration aid; this option will not be supported in future releases.

EXAMPLES

Components of two SunLink products, **/usr/sunlink/dni/dnilogind** (the DECNET analog of **rlogind**(8C) and **/usr/sunlink/x25/x29** (the OSI analog of **rlogind**), are known to need this wrapper. Typical usage is:

```
example% cd /usr/sunlink/dni
example% mv dnilogind .dnilogind
example% cat > dnilogind
#!/bin/sh
/usr/etc/setsid –b /usr/sunlink/dni/.dnilogind "$@"
^D
example% chmod +x dnilogind
```

SEE ALSO

setsid(2V)

IEEE Std 1003.1-1988

NAME

showfh – print full pathname of file from the NFS file handle

SYNOPSIS

/usr/etc/showfh *server_name num1 num2 ... num8*

DESCRIPTION

showfh prints the full path name of the file on the server for the given file handle (*num1 ... num8*). *server_name* is the server from where the client got this file handle. *num1 ... num8* are the file handle numbers represented in hexadecimal notation.

The **showfhd** daemon should be running on the NFS servers to answer **showfh** requests. If it cannot find the file corresponding to the given file handle, it prints a diagnostic message.

SEE ALSO

showfhd(8C)

BUGS

If the given NFS file handle is stale, then **showfh** may not print the name of the actual file. The inode for the file could have been allocated to some other file.

NAME

showfhd – showfh daemon run on the NFS servers

SYNOPSIS

/usr/etc/rpc.showfhd

DESCRIPTION

showfhd is the daemon which runs on the NFS servers and answers **showfh** requests. It provides the full path name for the given file handle. If it cannot find the file for the corresponding inode number, it returns an error message.

FILES

/etc/mtab	table of mounted file systems

SEE ALSO

find(1), showfh(8C)

NAME

showmount – show all remote mounts

SYNOPSIS

/usr/etc/showmount [**–ade**] [*hostname*]

AVAILABILITY

This program is available with the *Networking* software installation option. Refer to *Installing SunOS 4.1* for information on how to install optional software.

DESCRIPTION

showmount lists all the clients that have remotely mounted a filesystem from *host*. This information is maintained by the **mountd**(8C) server on *host*, and is saved across crashes in the file **/etc/rmtab**. The default value for *host* is the value returned by **hostname**(1).

OPTIONS

–a Print all remote mounts in the format:

hostname:*directory*

where *hostname* is the name of the client, and **directory** is the root of the file system that has been mounted.

–d List directories that have been remotely mounted by clients.

–e Print the list of exported file systems.

FILES

/etc/rmtab

SEE ALSO

hostname(1), **exports**(5), **exports**(5), **exportfs**(8), **mountd**(8C)

BUGS

If a client crashes, its entry will not be removed from the list until it reboots and executes '**umount –a**'.

NAME

shutdown – close down the system at a given time

SYNOPSIS

/usr/etc/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**(8), containing the time of shutdown, the instigator of the shutdown, and the reason. Then a terminate signal is sent to **init**, which brings the system down to single-user 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 Shut the system down in the manner of **fasthalt** (see **fastboot**(8)), so that when the system is rebooted, the file systems are not checked.

–h Execute **halt**(8).

–k Simulate shutdown of the system. Do not actually shut down the system.

–n Prevent the normal **sync**(2) before stopping.

–r Execute **reboot**(8).

FILES

/etc/nologin	tells login not to let anyone log in
/etc/xtab	list of remote hosts that have mounted this host

SEE ALSO

login(1), **sync**(2), **fastboot**(8), **halt**(8), **reboot**(8), **syslogd**(8)

BUGS

Only allows you to bring the system down between "now" and 23:59 if you use the absolute time for shutdown.

NAME

skyversion – print the SKYFFP board microcode version number

SYNOPSIS

/usr/etc/skyversion

DESCRIPTION

skyversion obtains from the SKYFFP board the Sky version number of the microcode currently loaded and prints the result on the standard output.

DIAGNOSTICS

The Sky version number operation code used to implement this command is not available for microcode releases earlier than Sky release 3.00. The result in this case is unpredictable and is either a nonmeaningful version number or a message indicating that no version number is available.

Meaningful version numbers are of the form *n.dd* where $n \geq 3$.

NAME

spray – spray packets

SYNOPSIS

/usr/etc/spray [**–c** *count*] [**–d** *delay*] [**–i** *delay*] [**–l** *length*] *host*

AVAILABILITY

This program is available with the *Networking* software installation option. Refer to *Installing SunOS 4.1* for information on how to install optional software.

DESCRIPTION

spray sends a one-way stream of packets to *host* using RPC, and reports how many were received, as well as the the transfer rate. The *host* argument can be either a name or an internet address.

OPTIONS

–c *count*
: Specify how many packets to send. The default value of *count* is the number of packets required to make the total stream size 100000 bytes.

–d *delay*
: Specify how many microseconds to pause between sending each packet. The default is 0.

–i *delay*
: Use ICMP echo packets rather than RPC. Since ICMP automatically echos, this creates a two way stream.

–l *length*
: The *length* parameter is the numbers of bytes in the Ethernet packet that holds the RPC call message. Since the data is encoded using XDR, and XDR only deals with 32 bit quantities, not all values of *length* are possible, and **spray** rounds up to the nearest possible value. When *length* is greater than 1514, then the RPC call can no longer be encapsulated in one Ethernet packet, so the *length* field no longer has a simple correspondence to Ethernet packet size. The default value of *length* is 86 bytes (the size of the RPC and UDP headers).

SEE ALSO

icmp(4P), **ping**(8C), **sprayd**(8C)

Installing SunOS 4.1

NAME

sprayd, rpc.sprayd – spray server

SYNOPSIS

/usr/etc/rpc.sprayd

AVAILABILITY

This program is available with the *Networking* software installation option. Refer to *Installing SunOS 4.1* for information on how to install optional software.

DESCRIPTION

rpc.sprayd is a server which records the packets sent by **spray**(8C), and sends a response to the originator of the packets. The **rpc.sprayd** daemon is normally invoked by **inetd**(8C).

SEE ALSO

inetd(8C), **spray**(8C)

Installing SunOS 4.1

NAME

start_applic – generic application startup procedures

SYNOPSIS

/usr/etc/start_applic

AVAILABILITY

Available only on Sun 386i systems running a SunOS 4.0.*x* release or earlier. Not a SunOS 4.1 release feature.

DESCRIPTION

start_applic is a short generic shell script that can be copied or symbolically linked into either **/vol/local/bin/***application* or **/usr/local/bin/***application*. When invoked as *application*, an application installed as described below will be correctly invoked on systems of any supported processor architecture. Installing **start_applic** (or a customized version of it) in one of these locations ensures that no user's or system's environment needs to be modified just to run the application. Applications are stored in a single tree which is not shared with any other applications. This tree may be available on different systems in different places; if the application needs to reference its distribution tree, this should be determined from the *application*_ROOT environment variable.

The application startup script arranges that the PATH and *application*_ROOT environment variables are set correctly while the application is running. If the application's distribution tree (placed into **/vol/***application* or **/usr/local/***application*) does not have an executable binary with the name of the application (for example, **/vol/***application*/**bin.***arch*/*application*), then **start_applic** can not be used, and a customized application startup script must be used instead. Such scripts must also allow users to invoke the application from systems of any architecture, without requiring them to customize their own environments.

Note that there are two contrasting models of software installation. The **heterogeneous model** assumes general availability of the software, and solves the "which binaries to use" problem with no administrative overhead. The **homogeneous model** assumes very limited availability of software, requires administrative procedures to ensure that **/usr/local** only contains binaries of the local architecture, and does not really account for networked installations. It is easier to add support for additional architectures using a heterogeneous network model of software installation from the beginning.

Heterogeneous Networked Installations

Applications available on the network are available through **/vol/***application* and exported either to all systems or just to selected ones, as licensing restrictions allow. The export point is **/export/vol/***application*, which is a symbolic link to the actual installation point, typically the **/files/vol/***application* directory. All subdirectories not explicitly tagged with a processor architecture are shared among all processor architectures; thus while the **.../bin.sun386** and **.../lib.sun386** subdirectories contain, respectively, binaries and libraries executable only on systems of the Sun386i architecture, the **.../bin** directory contains executables that run on any architecture (typically using an interpreter such as **/bin/sh**), and the **.../etc** directory only contains sharable configuration files.

Homogeneous Single Machine Installations

Applications available only on a specific machine and its boot clients of the same architecture are installed into **/usr/local/***application*. This directory supports only a single architecture, so **/usr/local/***application*/**bin** contains binaries executable only on the local architecture, and **/usr/local/***application*/**lib** contains libraries executable only on the local architecture. Any sharable files are grouped in **/usr/local/***application*/**share**.

To install an application onto a boot server to serve boot clients with other architectures, place the application in **/usr/local/***application* on the clients, as described above. The installation point (on the server) for application binaries of architecture *arch* is **/export/local/***arch*/*application*. When the architecture is the server architecture, this case is identical with the one above.

Other Installations

Smaller applications (of only one or two files) may be installed into the appropriate **/vol/local/bin.***arch* directory, or possibly into **/export/local/***arch*/**bin**. These directories are in user's default paths, so the application does not need to be registered using **start_applic**.

FILES

/files<n>/vol/*application*
/export/vol/*application*
/vol/*application*
/vol/*application***/bin.***arch***/application**
/usr/local/*application*
/export/local/*arch***/***application*

SEE ALSO

auto.vol(5), **exports**(5), **automount**(8), **exportfs**(8)

Sun386i SNAP Administration

Sun386i Advanced Administration

NAME

statd, rpc.statd – network status monitor

SYNOPSIS

/usr/etc/rpc.statd

DESCRIPTION

statd is an intermediate version of the status monitor. It interacts with **lockd**(8C) to provide the crash and recovery functions for the locking services on NFS.

FILES

/etc/sm
/etc/sm.bak
/etc/state

SEE ALSO

statmon(5), **lockd**(8C)

BUGS

The crash of a site is only detected upon its recovery.

NAME

sticky – mark files for special treatment

DESCRIPTION

The *sticky bit* (file mode bit 01000, see **chmod**(2V)) is used to indicate special treatment of certain files and directories. A directory for which the sticky bit is set restricts deletion of files it contains. A file in a sticky directory may only be removed or renamed by a user who has write permission on the directory, and either owns the file, owns the directory, or is the super-user. This is useful for directories such as **/tmp**, which must be publicly writable, but should deny users permission to arbitrarily delete or rename the files of others.

If the sticky bit is set on a regular file and no execute bits are set, the system's page cache will not be used to hold the file's data. This bit is normally set on swap files of diskless clients so that accesses to these files do not flush more valuable data from the system's cache. Moreover, by default such files are treated as swap files, whose inode modification times may not necessarily be correctly recorded on permanent storage.

Any user may create a sticky directory. See **chmod** for details about modifying file modes.

BUGS

mkdir(2V) will not create a file with the sticky bit set.

FILES

/tmp

SEE ALSO

chmod(1V), **chmod**(2V), **chown**(2V), **mkdir**(2V)

NAME

sundiag – system diagnostics

SYNOPSIS

/usr/diag/sundiag/sundiag [**–Cmt**] [**–k** *kernel_name*] [**–o** *saved_options_file*]
[*generic_tool_arguments*]

AVAILABILITY

This program is available with the *User Diagnostics* software installation option. Refer to *Installing SunOS 4.1* for information on how to install optional software.

DESCRIPTION

sundiag is a diagnostic facility that tests the functionality of the operating system and reports its findings. It can also be used to report the hardware configuration as detected by the system.

You must be **root** to use **sundiag**.

When run on the console monitor, **sundiag** takes full advantage of the *SunView 1* windowing environment. There are four subwindows:

- A control panel for displaying the discovered hardware configuration and manipulating of the numerous test parameters and options.
- A test status panel which shows the test results.
- A console window which is used to display messages.
- A performance monitor.

There are also some popup frames, including a text frame for viewing **sundiag** and system log files.

When executed from a terminal, **sundiag** uses **curses**(3V) to simulate each subwindow on the screen.

sundiag consists of **sundiag**, along with several binary modules and executable files containing the actual test code, all of which reside in **/usr/diag/sundiag**.

OPTIONS

–C Redirect the console output from any existing console window to the **sundiag** console subwindow.

–m Create a device file for all devices found during the kernel probe. **sundiag** uses the same major/minor device numbers and permissions declared in **/dev/MAKEDEV**.

–t Run **sundiag** on a terminal.

–k *kernel_name*
Specify the customized kernel name that was used to boot up the system. The default kernel name is **/vmunix**. Since the **rstatd**(8C) that the performance monitor requires is hard-wired to use **/vmunix** as the kernel name, the performance monitor is disabled when this option is specified.

–o *saved_options_file*
Use the *saved_options_file* to restore options. The default option file is **.sundiag**. **.sundiag** is used if the **–o** option is not used and if the default file exists.

generic_tool_arguments
Refer to **sunview**(1) for examples of generic tool arguments that may be used with **sundiag**.

FILES

/var/adm/sundiaglog/options/.sundiag	start-up option file
/usr/diag/sundiag/.usertest	user-defined test description file

SEE ALSO

sunview(1), **curses**(3V), **rstatd**(8C)

Installing SunOS 4.1
Sundiag User's Guide

NAME

suninstall – install and upgrade the SunOS operating system

SYNOPSIS

/usr/etc/install/suninstall

DESCRIPTION

suninstall is a forms-based subsystem for installing and upgrading the SunOS operating system. Unlike previous installation subsystems, **suninstall** does not require recapitulation of an interrupted procedure; you can pick up where you left off. A new invocation of **suninstall** displays the saved information and offers the user an opportunity to make any needed alterations before it proceeds.

Note: **suninstall** only exists in the mini-root and should only be invoked from there (see *Installing SunOS 4.1*).

suninstall allows installation of the operating system onto any system configuration, be it standalone, dataless, a homogeneous file server, or a heterogeneous server. It installs the various versions of the operating system needed by clients on a heterogeneous file server, from any Sun distribution media format. The number of different system versions that can be installed is only limited to the disk space available.

After the initial installation, the suninstall utility program **add_client**(8) adds clients while the server is running in multiuser mode. The suninstall **add_services**(8) program converts a standalone system or server into a heterogeneous file server, without rebooting, while the system is running in multiuser mode. To remove a diskless client, use the suninstall **rm_client**(8) program in multiuser mode.

To abort the installation procedure, use the interrupt character (typically CTRL-C).

USAGE

Refer to *Installing SunOS 4.1* for more information on the various menus and selections.

FILES

/usr/etc/install	directory containing installation programs and scripts
/usr/etc/install/xdrtoc	subsystem utility program
/etc/install	directory containing suninstall data files

SEE ALSO

add_client(8), **add_services**(8), **extract_unbundled**(8), **rm_client**(8)

Installing SunOS 4.1

NOTES

It is advisable to exit **suninstall** through the exit options from the **suninstall** menus.

NAME

swapon – specify additional device for paging and swapping

SYNOPSIS

/usr/etc/swapon –a

/usr/etc/swapon *name...*

DESCRIPTION

swapon specifies additional devices or files on which paging and swapping are to take place. The system begins by swapping and paging on only a single device so that only one disk is required at bootstrap time. Calls to **swapon** normally occur in the system multi-user initialization file **/etc/rc** making all swap devices available, so that the paging and swapping activity is interleaved across several devices.

The second form gives individual block devices or files as given in the system swap configuration table. The call makes only this space available to the system for swap allocation.

Note: "swap files" made with **mkfile**(8) can be used as swap areas over NFS.

OPTIONS

–a Make available all devices of type **swap** in **/etc/fstab**. Using **swapon** with the **–a** option is the normal usage.

FILES

/dev/sd?b	
/dev/xy?b	
/dev/xd?b	normal paging devices
/etc/fstab	
/etc/rc	

SEE ALSO

swapon(2), **fstab**(5), **init**(8), **mkfile**(8)

BUGS

There is no way to stop paging and swapping on a device. It is therefore not possible to make use of devices which may be dismounted during system operation.

NAME

sys-config – configure a system or administer configuration information

SYNOPSIS

/usr/etc/install/sys-config

DESCRIPTION

sys-config "unpacks" a machine and sets up its configuration. **sys-config** automatically runs when a pre-installed system is booted for the first time. It should not be run by hand. Instead, run **sys-unconfig**(8) to return the system to its pre-installed state. Then, reboot system, which will run **sys-config** automatically.

A system's configuration consists of hostname, Network Interface Service (NIS) domain name, timezone and IP address.

sys-config does the following:

- Edits the **/etc/hosts** with the correct hostname and IP address.
- Sets the hostname in **/etc/rc.boot**.
- Sets the domainname in **/etc/rc.single**.
- Sets the **/usr/lib/zoneinfo/localtime** file.
- Enables the Network Information Service (NIS) if the NIS service was requested.

When **sys-config** is finished, it prompts for a system reboot.

The default answer to any particular question is the current value of that configuration parameter. Parameters that have not changed can be quickly skipped over to get to the one that should be changed by typing a RETURN.

sys-config is potentially a dangerous utility and can be run only by the super-user.

FILES

/etc/hosts
/usr/lib/zoneinfo/localtime
/usr/etc/install/sys_info

SEE ALSO

sys-unconfig(8)

NOTES

The Network Information Service (NIS) was formerly known as Sun Yellow Pages (YP). The functionality of the two remains the same; only the name has changed.

NAME

sys-unconfig – undo a system's configuration

SYNOPSIS

/usr/etc/install/sys-unconfig

DESCRIPTION

sys-unconfig packs up a machine to make it ready to be configured again.

It restores a systems's configuration to an "as-manufactured" state. A system's configuration consists of hostname, Network Interface Service (NIS) domain name, timezone and IP address.

sys-unconfig does the following:

- Restores the default **/etc/hosts** file.
- Removes the default hostname in **/etc/hostname.??[0-9]**.
- Removes the default domainname in **/etc/defaultdomain**.
- Removes the default **/usr/lib/zoneinfo/localtime** file.
- Disables the Network Information Service (NIS) if the NIS service was requested.

When **sys-unconfig** is finished, it will prompt for a system shutdown.

sys-unconfig is potentially a dangerous utility and can only be run by the super-user.

FILES

/etc/hosts
/usr/lib/zoneinfo/localtime
/usr/etc/install/sys_info

SEE ALSO

sys-config(8)

NOTES

The Network Information Service (NIS) was formerly known as Sun Yellow Pages (YP). The functionality of the two remains the same; only the name has changed.

NAME

syslogd – log system messages

SYNOPSIS

/usr/etc/syslogd [**–d**] [**–f***configfile*] [**–m** *interval*]

DESCRIPTION

syslogd reads and forwards system messages to the appropriate log files and/or users, depending upon the priority of a message and the system facility from which it originates. The configuration file **/etc/syslog.conf** (see **syslog.conf**(5)) controls where messages are forwarded. **syslogd** logs a mark (timestamp) message every *interval* minutes (default 20) at priority **LOG_INFO** to the facility whose name is given as **mark** in the **syslog.conf** file.

A system message consists of a single line of text, which may be prefixed with a priority code number enclosed in angle-brackets (**<>**); priorities are defined in **sys/syslog.h**.

syslogd reads from the **AF_UNIX** address family socket **/dev/log**, from an Internet address family socket specified in **/etc/services**, and from the special device **/dev/klog** (for kernel messages).

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 file **/etc/syslog.pid**, if possible, containing its process ID (PID).

Sun386i DESCRIPTION

syslogd translates messages using the databases specified on an optional line in the **syslog.conf** as indicated with a **translate** entry.

The format of these databases is described in **translate**(5).

OPTIONS

–d	Turn on debugging.
–f*configfile*	Specify an alternate configuration file.
–m *interval*	Specify an interval, in minutes, between mark messages.

FILES

/etc/syslog.conf	configuration file
/etc/syslog.pid	process ID
/dev/log	AF_UNIX address family datagram log socket
/dev/klog	kernel log device
/etc/services	network services database

SEE ALSO

logger(1), **syslog**(3), **syslog.conf**(5), **translate**(5)

NAME

talkd, in.talkd – server for talk program

SYNOPSIS

/usr/etc/in.talkd

DESCRIPTION

talkd is a server used by the **talk**(1) program. It listens at the udp port indicated in the "talk" service description; see **services**(5). The actual conversation takes place on a tcp connection that is established by negotiation between the two machines involved.

SEE ALSO

talk(1), **services**(5), **inetd**(8C)

BUGS

The protocol is architecture dependent, and can not be relied upon to work between Sun systems and other machines.

NAME

telnetd, in.telnetd – TCP/IP TELNET protocol server

SYNOPSIS

/usr/etc/in.telnetd

AVAILABILITY

This program is available with the *Networking* software installation option. Refer to *Installing SunOS 4.1* for information on how to install optional software.

DESCRIPTION

telnetd is a server which supports the TCP/IP standard TELNET virtual terminal protocol. **telnetd** is invoked by the internet server (see **inetd**(8C)), normally for requests to connect to the TELNET port as indicated by the **/etc/services** file (see **services**(5)).

telnetd operates by allocating a pseudo-terminal device (see **pty**(4)) for a client, then creating a login process which has the slave side of the pseudo-terminal as its standard input, output, and error. **telnetd** manipulates the master side of the pseudo-terminal, implementing the TELNET protocol and passing characters between the remote client and the login process.

When a TELNET session is started up, **telnetd** sends TELNET options to the client side indicating a willingness to do *remote echo* of characters, to *suppress go ahead*, and to receive *terminal type information* from the remote client. If the remote client is willing, the remote terminal type is propagated in the environment of the created login process. The pseudo-terminal allocated to the client is configured to operate in "cooked" mode, and with **XTABS**, **ICRNL**, and **ONLCR** enabled (see **termio**(4)).

telnetd is willing to *do*: *echo*, *binary*, *suppress go ahead*, and *timing mark*. **telnetd** is willing to have the remote client *do*: *binary*, *terminal type*, and *suppress go ahead*.

SEE ALSO

telnet(1C)

Postel, Jon, and Joyce Reynolds, "Telnet Protocol Specification," RFC 854, Network Information Center, SRI International, Menlo Park, Calif., May 1983.

BUGS

Some TELNET commands are only partially implemented.

The TELNET protocol allows for the exchange of the number of lines and columns on the user's terminal, but **telnetd** doesn't make use of them.

Because of bugs in the original 4.2 BSD **telnet**(1C), **telnetd** performs some dubious protocol exchanges to try to discover if the remote client is, in fact, a 4.2 BSD **telnet**(1C).

Binary mode has no common interpretation except between similar operating systems

The terminal type name received from the remote client is converted to lower case.

The *packet* interface to the pseudo-terminal (see **pty**(4)) should be used for more intelligent flushing of input and output queues.

telnetd never sends TELNET *go ahead* commands.

telnetd can only support 64 pseudo-terminals.

NAME

tfsd – TFS daemon

SYNOPSIS

/usr/etc/tfsd

DESCRIPTION

tfsd is the daemon for the Translucent File Service (TFS). This daemon is started by **inetd**(8C) whenever a TFS request is made.

tfsd looks up a file by looking in the frontmost directory (see **tfs**(4S)). If the file is not found in this directory, **tfsd** follows the *searchlink* from the frontmost directory to the directory immediately behind it. **tfsd** continues to search for the file until one of the following conditions is met:

- The file is found in a directory.
- There are no more searchlinks to follow.
- A *whiteout* entry for the file is found.

The searchlinks and whiteout entries are specified in **.tfs_info** files.

FILES

.tfs_info	holds searchlink and whiteout entries

SEE ALSO

unwhiteout(1), **lsw**(1), **tfs**(4S), **mount_tfs**(8)

NAME

tftpd, in.tftpd – TCP/IP Trivial File Transfer Protocol server

SYNOPSIS

/usr/etc/in.tftpd [–s] [*homedir*]

Sun386i SYNOPSIS

/usr/etc/in.tftpd [–s] [–p] [*homedir*]

AVAILABILITY

This program is available with the *Networking* software installation option. Refer to *Installing SunOS 4.1* for information on how to install optional software.

DESCRIPTION

tftpd is a server that supports the TCP/IP Trivial File Transfer Protocol (TFTP). This server is normally started by **inetd**(8C) and operates at the port indicated in the **tftp** Internet service description in the **/etc/inetd.conf** file; see **inetd.conf**(5) for details.

Before responding to a request, the server attempts to change its current directory to *homedir*; the default value is **/tftpboot**.

Sun386i DESCRIPTION

The **tftpd** daemon acts as described above, except that it will perform certain filename mapping operations unless instructed otherwise by the **–p** command line argument or when operating in a secure environment. This mapping affects only TFTP boot requests and will not affect requests for existing files.

The semantics of the changes are as follows. Only filenames of the format *ip-address* or *ip-address.arch*, where *ip-address* is the IP address in hex, and *arch* is the hosts's architecture (as returned by the **arch**(1) command), that do not correspond to files in **/tftpboot**, are mapped. If the address is known through a Network Interface Service (NIS) lookup, any file of the form **/tftpboot/ip-address*** (with or without a suffix) is returned. If there are multiple such files, any one may be returned. If the *ip-address* is unknown (that is if the **ipalloc (8C)** service says the name service does not know the address), the filename is mapped as follows: Names without the *arch* suffix are mapped into the name **pnp.SUN3**, and names with the suffix are mapped into **pnp.** *arch*. That file is returned if it exists.

OPTIONS

–s Secure. When specified, the directory change must succeed; and the daemon also changes its root directory to *homedir*.

The use of **tftp** does not require an account or password on the remote system. Due to the lack of authentication information, **tftpd** will allow only publicly readable files to be accessed. Files may be written only if they already exist and are publicly writable. Note: this extends the concept of "public" to include all users on all hosts that can be reached through the network; this may not be appropriate on all systems, and its implications should be considered before enabling this service.

tftpd runs with the user ID (UID) and group ID (GID) set to **–2**, under the assumption that no files exist with that owner or group. However, nothing checks this assumption or enforces this restriction.

Sun386i OPTIONS

–p Disable pnp entirely. Do not map filenames.

Sun386i FILES

/tftpboot/* filenames are IP addresses

SEE ALSO

tftp(1C) **inetd**(8C), **ipallocd**(8C), **netconfig**(8C)

Sollins, K.R., *The TFTP Protocol (Revision 2)*, RFC 783, Network Information Center, SRI International, Menlo Park, Calif., June 1981.

NOTES

The Network Information Service (NIS) was formerly known as Sun Yellow Pages (YP). The functionality of the two remains the same; only the name has changed.

Sun386i WARNINGS

A request for an *ip-address* from a Sun-4 can be satisfied by a file named *ip-address*.**386** for compatibility with some early Sun-4 PROM monitors.

NAME

tic – terminfo compiler

SYNOPSIS

tic [**–v**[*n*]] [**–c**] *filename*

AVAILABILITY

This command is available with the *System V* software installation option. Refer to *Installing SunOS 4.1* for information on how to install optional software.

DESCRIPTION

tic compiles a **terminfo**(5V) source file into the compiled format. The results are placed in the directory **/usr/share/lib/terminfo**. The compiled format is used by the **curses**(3V) library.

Each entry in the file describes the capabilities of a particular terminal. When a **use**=*entry* field is given in a terminal entry, **tic** reads in the binary (compiled) description of the indicated *entry* from **/usr/share/lib/terminfo** to duplicate the contents of that entry within the one being compiled. However, if an *entry* by that name is specified in *filename*, the entry in that source file is used first. Also, if a capability is defined in both entries, the definition in the current entry's source file is used.

If the environment variable **TERMINFO** is set, that directory is searched and written to instead of **/usr/share/lib/terminfo**.

OPTIONS

–v[*n*]
: Verbose. Display trace information on the standard error. The optional integer argument is a number from 1 to 10, inclusive, indicating the desired level of detail. If *n* is omitted, the default is 1.

–c
: Only check *filename* for errors. Errors in **use**= links are not detected.

FILES

/usr/share/lib/terminfo/?/* compiled terminal description data base

SEE ALSO

fork(2V), **curses**(3V), **curses**(3V), **malloc**(3V), **term**(5), **terminfo**(5V)

BUGS

Total compiled entries cannot exceed 4096 bytes. The name field cannot exceed 1024 bytes.

When the **–c** option is used, duplicate terminal names will not be diagnosed; however, when **–c** is not used, they will be.

For backward compatibility, cancelled capabilities will not be marked as such within the terminfo binary unless the entry name has a '+' within it. Such terminal names are only used for inclusion with a **use**= field, and typically aren't used for actual terminal names.

DIAGNOSTICS

Most diagnostic messages produced by **tic** are preceded with the approximate line number and the name of the entry being processed.

mkdir *name* **returned bad status**
: The named directory could not be created.

File does not start with terminal names in column one
: The first thing seen in the file, after comments, must be the list of terminal names.

Token after a seek(2) not NAMES
: Somehow the file being compiled changed during the compilation.

Not enough memory for use_list element
Out of memory
Not enough free memory was available (**malloc**(3V) failed).

Can't open *filename*
The named file could not be opened or created.

Error in writing *filename*
The named file could not be written to.

Can't*link* **filename** *to* **filename**
A link failed.

Error in re-reading compiled *filename*
The compiled file could not be read back in.

Premature EOF
The current entry ended prematurely.

Backspaced off beginning of line
This error indicates something wrong happened within **tic**.

Unknown Capability – *filename*
The named invalid capability was found within the file.

Wrong type used for capability ...
For example, a string capability was given a numeric value.

Unknown token type
Tokens must be followed by '@' to cancel, ',' for booleans, '#' for numbers, or '=' for strings.
name: **bad term name**
Line *n*: **Illegal terminal name** – *name*
Terminal names must start with a letter or digit
The given name was invalid. Names must not contain white space or slashes, and must begin with a letter or digit.

name: **terminal name too long.**
An extremely long terminal name was found.

name: **terminal name too short.**
A one-letter name was found.

name **defined in more than one entry. Entry being used is** *name* .
An entry was found more than once.

Terminal name *name* **synonym for itself**
A name was listed twice in the list of synonyms.

At least one synonym should begin
At least one of the names of the terminal should begin with a letter.

Illegal character – **c**
The given invalid character was found in the input file.

Newline in middle of terminal name
The trailing comma was probably left off of the list of names.

Missing comma
A comma was missing.

Missing numeric value
The number was missing after a numeric capability.

NULL string value
The proper way to say that a string capability does not exist is to cancel it.

Very long string found. Missing comma?
Self-explanatory.

Unknown option. Usage is:
An invalid option was entered.

Too many file names. Usage is:
Self-explanatory.

name **non-existent or permission denied**
The given directory could not be written into.

name **is not a directory**
Self-explanatory.

name: **Permission denied**
Access denied.

name: **Not a directory**
tic wanted to use the given name as a directory, but it already exists as a file

SYSTEM ERROR!! Fork failed!!!
A **fork**(2V) failed.

Error in following up use-links.
Either there is a loop in the links or they reference non-existent terminals. The following is a list of the entries involved:
A **terminfo**(5V) entry with a **use**=*name* capability either referenced a non-existent terminal called *filename* or *filename* somehow referred back to the given entry.

NAME

tnamed, in.tnamed – TCP/IP Trivial name server

SYNOPSIS

/usr/etc/in.tnamed [–v]

DESCRIPTION

tnamed is a server that supports the TCP/IP Name Server Protocol. The name server operates at the port indicated in the "name" service description (see **services**(5)), and is invoked by **inetd**(8C) when a request is made to the name server.

Two known clients of this service are the MIT PC/IP software the Bridge boxes.

OPTIONS

–v Invoke the daemon in verbose mode.

SEE ALSO

uucp(1C), **services**(5), **inetd**(8C)

Postel, Jon, *Internet Name Server*, IEN 116, SRI International, Menlo Park, California, August 1979.

BUGS

The protocol implemented by this program is obsolete. Its use should be phased out in favor of the Internet Domain protocol. See **named**(8C).

NAME

trpt – transliterate protocol trace

SYNOPSIS

/usr/etc/trpt [**–afjst**] [**–p***hex-address*] [*system* [*core*]]

DESCRIPTION

trpt interrogates the buffer of TCP trace records created when a socket is marked for "debugging" (see **getsockopt**(2)), and prints a readable description of these records. When no options are supplied, **trpt** prints all the trace records found in the system grouped according to TCP connection protocol control block (PCB). The following options may be used to alter this behavior.

OPTIONS

–a In addition to the normal output, print the values of the source and destination addresses for each packet recorded.

–f Follow the trace as it occurs, waiting a short time for additional records each time the end of the log is reached.

–j Just give a list of the protocol control block addresses for which there are trace records.

–s In addition to the normal output, print a detailed description of the packet sequencing information.

–t In addition to the normal output, print the values for all timers at each point in the trace.

–p *hex-address*
Show only trace records associated with the protocol control block, the address of which follows.

The recommended use of **trpt** is as follows. Isolate the problem and enable debugging on the **socket**(s) involved in the connection. Find the address of the protocol control blocks associated with the sockets using the **–A** option to **netstat**(8C). Then run **trpt** with the **–p** option, supplying the associated protocol control block addresses. The **–f** option can be used to follow the trace log once the trace is located. If there are many sockets using the debugging option, the **–j** option may be useful in checking to see if any trace records are present for the socket in question.

If debugging is being performed on a system or core file other than the default, the last two arguments may be used to supplant the defaults.

FILES

/vmunix
/dev/kmem

SEE ALSO

getsockopt(2), **netstat**(8C)

DIAGNOSTICS

no namelist When the system image does not contain the proper symbols to find the trace buffer; others which should be self explanatory.

BUGS

Should also print the data for each input or output, but this is not saved in the trace record.

The output format is inscrutable and should be described here.

NAME

ttysoftcar – enable/disable carrier detect

SYNOPSIS

ttysoftcar [**–y** | **–n**] *tty* ...

ttysoftcar –a

DESCRIPTION

For each *tty* specified **ttysoftcar** changes the carrier detect flag using the **TIOCSSOFTCAR ioctl()** request (see **tty**(4)). If the **–a** option is specified, **ttysoftcar** sets all tty's in the **/etc/ttytab** file to the carrier detection mode specified by their status field. If this field is set to **local**, software carrier detection is turned on. If this field is set to anything other than **local**, as is usually the case for modems, software carrier detection is turned off. **ttysoftcar** ignores devices in the **/etc/ttytab** file which do not exist.

If no options are specified, **ttysoftcar** returns the current status for *tty*. This status is reported as **y** or **n**.

OPTIONS

–a Reset ttys to appropriate values based on the status field of the **/etc/ttytab** file.

–y Turn on software carrier detect.

–n Turn off software carrier detect. Use hardware carrier detect.

SEE ALSO

termio(4), **zs**(4S), **ttytab**(5)

NAME

tunefs – tune up an existing file system

SYNOPSIS

/usr/etc/tunefs [**–a** *maxcontig*] [**–d** *rotdelay*] [**–e** *maxbpg*] [**–m** *minfree*] *special* | *filesystem*

DESCRIPTION

tunefs is designed to change the dynamic parameters of a file system which affect the layout policies. The parameters which are to be changed are indicated by the OPTIONS given below:

OPTIONS

–a *maxcontig*

This specifies the maximum number of contiguous blocks that will be laid out before forcing a rotational delay (see **–d** below). The default value is one, since most device drivers require an interrupt per disk transfer. Device drivers that can chain several buffers together in a single transfer should set this to the maximum chain length.

–d *rotdelay*

This specifies 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.

–e *maxbpg*

This 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 this value is set to about one quarter of the total blocks in a cylinder group. The intent is 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*

This value specifies the percentage of space held back from normal users; the minimum free space threshold. The default value used is 10%. This value can be set to zero, however up to a factor of three in throughput will be lost over the performance obtained at a 10% threshold. Note: if the value is raised above the current usage level, users will be unable to allocate files until enough files have been deleted to get under the higher threshold.

SEE ALSO

fs(5), **dumpfs**(8), **mkfs**(8), **newfs**(8)

System and Network Administration

BUGS

This program should work on mounted and active file systems. Because the super-block is not kept in the buffer cache, the program will only take effect if it is run on dismounted file systems; if run on the root file system, the system must be rebooted.

NAME

tzsetup – set up old-style time zone information in the kernel

SYNOPSIS

/usr/etc/tzsetup

DESCRIPTION

tzsetup attempts to find the offset from GMT and old-style Daylight Savings Time correction type (see **gettimeofday**(2)) that most closely matches the default time zone for the machine, and to pass this information to the kernel with a **settimeofday** ( ) call (see **gettimeofday**(2)). This is necessary if programs built under releases of SunOS prior to 4.0 are to be run; those programs get time zone information from the kernel using **gettimeofday**.

If it cannot find the offset from GMT, the offset is set to 0; if it cannot find the Daylight Savings Time correction type, it is set to DST_NONE, indicating that no Daylight Savings Time correction is to be performed.

DIAGNOSTICS

tzsetup: Can't open /usr/share/lib/zoneinfo/localtime: *reason*
The time zone file for the current time zone could not be opened.

tzsetup: Error reading /usr/lib/zoneinfo/localtime: *reason*
The time zone file for the current time zone could not be read.

tzsetup: Two or more time zone types are equally valid — no DST selected
There were two or more Daylight Savings Time correction types that generated results that were equally close to the correct results. None of them was selected. Programs built under versions of SunOS prior to 4.0 may not convert dates correctly.

tzsetup: No old-style time zone type is valid — no DST selected
None of the Daylight Savings Time correction types generated results that were in any way correct; none of them was selected. Programs built under versions of SunOS prior to 4.0 may not convert dates correctly.

tzsetup: Warning: No old-style time zone type is completely valid
None of the Daylight Savings Time correction types generated results that were completely correct; the best of them was selected. Programs built under versions of SunOS prior to 4.0 may not convert dates correctly.

tzsetup: Can't set time zone
tzsetup was run by a user other than the super-user; only the super-user may change the kernel's notion of the current time zone.

SEE ALSO

gettimeofday(2), **tzfile**(5), **zic**(8)

NAME

uid_allocd, gid_allocd – UID and GID allocator daemons

SYNOPSIS

/usr/etc/rpc.uid_allocd
/usr/etc/rpc.gid_allocd

AVAILABILITY

Available only on Sun 386i systems running a SunOS 4.0.*x* release or earlier. Not a SunOS 4.1 release feature.

DESCRIPTION

The UID (or GID) allocator will temporarily allocate an unused UID (or GID) for use by account administration tools. It maintains a cache of UIDs (GIDs) that have been allocated by potentially multiple tools (or instances of tools) in a distributed system, so that they can create accounts (or groups) concurrently. It also provides the ability to safely enter a UID (GID) into the cache which was allocated using some other method, such as manually by an administrator; and the ability to delete entries from the cache. Entries in this cache persist for at least an hour even through system crashes.

These allocators are available on the system which contains the master copy of the list of UIDs (or GID). Since this list is currently maintained using the Network Interface Service (NIS), the service is available on the master of the **passwd.byuid** (**group.bygid**) NIS map. The service could be provided using a UID database service other than the NIS service.

This implementation uses DES authentication (the Sun Secure RPC protocol) to restrict access to this function. The only clients privileged to allocate UIDs (GIDs) are those whose net IDs are in the *accounts* group (fixed at GID 11). All machine IDs are allowed to allocate UIDs (GIDs).

If the file **/etc/ugid_alloc.range** exists, the allocator only allocates UIDs (GIDs) in the range listed there. This feature is intended to be used by sites which have multiple NIS domains on their networks; each NIS domain would be assigned a unique range of UIDs (GIDs). If the file exists, and the local NIS domain is not explicitly assigned a unique range of UIDs or GID, none will be allocated. Without a mechanism to ensure that UIDs are uniquely assigned between NIS domains that share resources, normal NFS security mechanisms (excluding Secure NFS) may fail to serve as an advisory security mechanism. Common alternative methods for ensuring UID uniqueness include using a function of some preexisting identifier such as an employee number, or using a single NIS domain for the entire site.

FILES

/var/yp/*domainname***/passwd.byuid.{dir,pag}**
/var/yp/*domainname***/group.bygid.{dir,pag}**
/var/yp/*domainname***/netid.byname.{dir,pag}**
/etc/uid_alloc.cache
/etc/gid_alloc.cache
/etc/ugid_alloc.range
/usr/include/rpcsvc/uid_alloc.x
/usr/include/rpcsvc/gid_alloc.x

SEE ALSO

snap(1), **ugid_alloc.range**(5), **logintool**(8)

BUGS

Using UID (GID) ranges does not solve the problem that two different machines, or groups of machines, may assign different meaning to a given UID (GID).

The current implementation of the daemon is tuned towards small lists of active UIDs (GIDs), both in the NIS service and in the cache it maintains.

NOTES

The Network Information Service (NIS) was formerly known as Sun Yellow Pages (YP). The functionality of the two remains the same; only the name has changed.

NAME

unadv – unadvertise a Remote File Sharing resource

SYNOPSIS

unadv *resource*

AVAILABILITY

This program is available with the *RFS* software installation option. Refer to *Installing SunOS 4.1* for information on how to install optional software.

DESCRIPTION

unadv unadvertises a Remote File Sharing (RFS) *resource*, which is the advertised symbolic name of a local directory, by removing it from the advertised information on the domain name server. **unadv** prevents subsequent remote mounts of that resource. It does not affect continued access through existing remote or local mounts.

An administrator at a server can unadvertise only those resources that physically reside on the local machine. A domain administrator can unadvertise any resource in the domain from the primary name server by specifying *resource* name as *domain.resource*. A domain administrator should only unadvertise another hosts resources to clean up the domain advertise table when that host goes down. Unadvertising another host's resource changes the domain advertise table, but not the host advertise table.

This command is restricted to the super-user.

If *resource* is not found in the advertised information, an error message will be sent to standard error.

SEE ALSO

adv(8), **fumount**(8), **nsquery**(8)

NAME

unconfigure – reset the network configuration for a Sun386i system

SYNOPSIS

/usr/etc/unconfigure [**–y**]

AVAILABILITY

Available only on Sun 386i systems running a SunOS 4.0.*x* release or earlier. Not a SunOS 4.1 release feature.

DESCRIPTION

unconfigure restores most of the system configuration and status files to the state they were in when delivered by Sun Microsystems, Inc. It also deletes all user accounts (including home directories), Network Interface Service (NIS) information, and any diskless client configurations that were set up.

After running **unconfigure,** a system halts. Rebooting it to multi-user mode at this point will start automatic system installation.

unconfigure is intended for use in the following situations:

- As one of the final steps in Software Manufacturing.
- In systems being set up with temporary configurations, holding no user accounts or diskless clients. These will occur during demonstrations and evaluation trials.
- To allow systems that had been used as standalones to be upgraded to join a network in a role other than as a master server. (See instructions later.)

unconfigure is potentially a dangerous utility; it does not work unless invoked by the super-user. As a warning, unless the –y option is passed, it will require confirmation that all user files and system software configuration information is to be deleted.

This utility is *not* recommended for routine use of any sort.

Resetting Temporary Configurations

If users need to set up and tear down configurations, **unconfigure** can be used to restore the system to an essentially as-manufactured state. The main concern here is that user accounts will be deleted, so this should not be done casually.

To reset a temporary configuration, just become the super-user and invoke **unconfigure.**

Upgrading Standalones to Network Clients

Systems that are going to be networked should be networked from the very first, if at all possible. This eliminates whole classes of compatibility problems, such as pathnames and (in particular) user account clashes.

Automatic system installation directly supports upgrading a single standalone system to an NIS master, and joining any number of unused systems (or systems upon which **unconfigure** has been run) into a network.

However, in the situation where standalone systems that have been used extensively are to be joined to a network, **unconfigure** can be used in conjunction with automatic system installation by a knowledgeable super-user to change a system's configuration from standalone to network client. This procedure is not recommended for use by inexperienced administrators.

The following procedure is not needed unless user accounts or other data need to be preserved; it is intended to ensure that every UID and GID is changed so as not to clash with those in use on the network. It must be applied to each system that is being upgraded from a standalone to a network client.

The procedure is as follows:

- Identify all accounts and files that you will want to save. If there are none, just run **unconfigure** and install the system on the network. Do not follow the remaining steps.
- Copy **/etc/passwd** to **/etc/passwd.bak.**

- Rename all the files (including home directories) so that they aren't deleted. (See **FILES** below.) These will probably be only in **/export/home**.
- Run **unconfigure** and install the system on the network.
- For each account listed in **/etc/passwd.bak** that you want to save, follow this procedure:
  - Create a new account on the network; if the UID and GID are the same as in **/etc/passwd.bak** on the standalone, then skip the next step. However, be sure that you do not make two different accounts with the same UID.
  - Use the '**chown –R**' command to change the ownership of the home directories.
  - You may need to rename the files you just chowned above, for example to ensure that they are the user's home directory. This may involve updating the **auto.home**(5) and **auto.home**(5) NIS maps, as well.
- Delete **/etc/passwd.bak**.

FILES

unconfigure deletes the following files, if they are present, replacing some of them with the distribution version if one is supposed to exist:

/etc/.rootkey	**/etc/ethers**	**/etc/localtime**	**/etc/publickey**
/etc/auto.home	**/etc/exports**	**/etc/net.conf**	**/etc/sendmail.cf**
/etc/auto.vol	**/etc/fstab**	**/etc/netmasks**	**/etc/syslog.conf**
/etc/bootparams	**/etc/group**	**/etc/networks**	**/etc/systems**
/etc/bootservers	**/etc/hosts**	**/etc/passwd**	**/single/ifconfig**
/var/sysex/*			

and all files in **/var/yp** except those distributed with the operating system.

unconfigure truncates all files in **/var/adm**. All user home directories in **/export/home** are deleted, except those for the default user account **users**, which is shipped with the operating system. All diskless client configuration information stored in **/export/roots**, **/export/swaps**, and **/export/dumps** is deleted.

SEE ALSO

chgrp(1), **find**(1), **group**(5), **passwd**(5) **adduser**(8), **chown**(8)

BUGS

More of the system configuration files should be reset.

This does not yet support taking a workstation off the network temporarily, for example, to take it home over the weekend for use as a standalone, or to move it to another network while traveling. This should be the default behavior.

The procedure for upgrading standalones to network clients should be automated; currently, only upgrading a standalone to a master server is automated.

NOTES

The Network Information Service (NIS) was formerly known as Sun Yellow Pages (YP). The functionality of the two remains the same; only the name has changed.

NAME

update – periodically update the super block

SYNOPSIS

/usr/etc/update

DESCRIPTION

update is a program that executes the **sync**(2) primitive every 30 seconds. This insures that the file system is fairly up to date in case of a crash. This command should not be executed directly, but should be executed out of the initialization shell command file.

SEE ALSO

sync(1), **sync**(2), **init**(8)

NAME

user_agentd – user agent daemon

SYNOPSIS

/usr/etc/rpc.user_agentd

AVAILABILITY

Available only on Sun 386i systems running a SunOS 4.0.*x* release or earlier. Not a SunOS 4.1 release feature.

DESCRIPTION

rpc.user_agentd is the remote service used by **snap**(1) to create, move, or delete home directories, and by the New User Accounts feature of **logintool**(8) to create new home directories. The user_agent daemon is normally invoked by **inetd**(8C), and runs on all non-diskless systems.

When creating a new home directory, the user_agent daemon executes the **copy_home**(8) script which resides in the home directory of the primary group to which a new user will be added.

SEE ALSO

snap(1), **copy_home**(8), **inetd**(8C), **logintool**(8)

NAME

uucheck – check the UUCP directories and Permissions file

SYNOPSIS

/usr/lib/uucp/uucheck [**–v**] [**–x** *debug_level*]

AVAILABILITY

This command is available with the *uucp* software installation option. Refer to *Installing SunOS 4.1* for information on how to install optional software.

DESCRIPTION

uucheck checks for the presence of the UUCP system required files and directories. It also checks for some obvious errors in the **Permissions** file (**/usr/lib/uucp/Permissions**).

Note: **uucheck** can only be used by the super-user or **uucp**.

OPTIONS

–v Give a detailed explanation of how the UUCP programs will interpret the **Permissions** file.

–x *debug_level*

Produce debugging output on the standard output. *debug_level* is a number between 0 and 9; higher numbers give more detailed information. 5, 7, and 9 are good numbers to try; they give increasing amounts of detail.

FILES

/etc/uucp/Systems
/etc/uucp/Permissions
/etc/uucp/Devices
/etc/uucp/Maxuuscheds
/etc/uucp/Maxuuxqts
/var/spool/uucp/*
/var/spool/locks/LCK*
/var/spool/uucppublic/*

SEE ALSO

uucp(1C), **uustat**(1C), **uux**(1C), **uucico**(8C), **uusched**(8C)

BUGS

The program does not check file/directory modes or some errors in the **Permissions** file such as duplicate login or machine name.

NAME

uucico – file transport program for the UUCP system

SYNOPSIS

/usr/lib/uucp/uucico [**–r** *role_number*] [**–x** *debug_level*] [**–i** *interface*] [**–d** *spool_directory*]
–s *system_name*

AVAILABILITY

This command is available with the *uucp* software installation option. Refer to *Installing SunOS 4.1* for information on how to install optional software.

DESCRIPTION

uucico is the file transport program for UUCP work file transfers. **uux**(1C) and **uucp**(1C) both queue jobs that will be transferred by **uucico**. It is normally started by the scheduler, **uusched (8C),** but can be started manually; this is done for debugging. For example, the script **Uutry** starts **uucico** with debugging turned on.

OPTIONS

–r *role_number*

Specify the role that **uucico** should perform. *role_number* is the digit 1 for master mode or 0 for slave mode (default). Master mode should be specified when **uucico** is started by a program or **cron**(8).

–x *debug_level*

Produce debugging output on the standard output. *debug_level* is a number between 0 and 9; higher numbers give more detailed information. 5, 7, and 9 are good numbers to try; they give increasing amounts of detail.

–i *interface*

Define the interface used with **uucico**. This interface only affects slave mode. Known interfaces are UNIX (default).

FILES

/etc/uucp/Systems
/etc/uucp/Permissions
/etc/uucp/Devices
/etc/uucp/Devconfig
/etc/uucp/Sysfiles
/etc/uucp/Maxuuxqts
/etc/uucp/Maxuuscheds
/var/spool/uucp/*
/var/spool/locks/LCK*
/var/spool/uucppublic/*

SEE ALSO

uucp(1C), **uustat**(1C), **uux**(1C), **cron**(8), **uusched**(8C)

NAME

uuclean – uucp spool directory clean-up

SYNOPSIS

/usr/lib/uucp/uuclean [**–m**] [**–d***directory*] [**–n***time*] [**–p***pre*]

DESCRIPTION

uuclean scans the spool directory for files with the specified prefix and deletes all those which are older than the specified number of hours.

OPTIONS

–d*directory*
: Clean the indicated spool directory.

–m
: Send mail to the owner of the file when it is deleted.

–n*time*
: Files whose age is more than *time* hours are deleted if the prefix test is satisfied (default time is 72 hours).

–p*pre*
: Scan for files with *pre* as the file prefix. Up to 10 **–p** arguments may be specified. A **–p** without any *pre* following deletes all files older than the specified time.

uuclean will typically be started by **cron**(8).

FILES

/usr/lib/uucp	directory with commands used by uuclean internally
/usr/lib/uucp/spool	spool directory

SEE ALSO

uucp(1C), **uux**(1C), **cron**(8)

NAME

uucleanup – UUCP spool directory clean-up

SYNOPSIS

/usr/lib/uucp/uucleanup [**–C***time*] [**–D***time*] [**–m***string*] [**–o***time*] [**–s***system*] [**–W***time*]
[**–x** *debug_level*] [**–X***time*]

AVAILABILITY

This command is available with the *uucp* software installation option. Refer to *Installing SunOS 4.1* for information on how to install optional software.

DESCRIPTION

uucleanup will scan the spool directories for old files and take appropriate action to remove them in a useful way:

- Inform the requestor of send/receive requests for systems that cannot be reached.
- Return mail, which cannot be delivered, to the sender.
- Delete or execute **rnews** for **rnews** type files (depending on where the news originated — locally or remotely).
- Remove all other files.

In addition, there is provision to warn users of requests that have been waiting for a given number of days (default 1 day). Note: **uucleanup** will process as if all option *time*s were specified to the default values unless *time* is specifically set.

This program is typically started by the shell **uudemon.cleanup**, which should be started by **cron**(8).

OPTIONS

–C*time* Remove any **C.** files that are at least *time* days old (default 7 days), and send appropriate information to the requestor.

–D*time* Remove any **D.** files that are at least *time* days old (default 7 days), and make an attempt to deliver mail messages and execute **rnews** when appropriate.

–m*string*
Include this line in the warning message generated by the –W option. The default line is '**See your local administrator to locate the problem**'.

–o*time* Delete other files that are more than *time* days old (default 2 days).

–s*system*
Execute for the spool directory for the remote system *system* only.

–W*time*
Send a mail message to be sent to the requestor warning about the delay in contacting the remote for any **C.** files that are *time* days old (default 1 day). The message includes the *JOBID*, and in the case of mail, the mail message. The administrator may include a message line telling whom to call to check the problem (**–m** option).

–x *debug_level*
Produce debugging output on the standard output. *debug_level* is a number between 0 and 9; higher numbers give more detailed information. 5, 7, and 9 are good numbers to try; they give increasing amounts of detail.

–X*time* Remove any **X.** files that are at least *time* days old (default 2 days). The **D.** files are probably not present (if they were, the **X.** could get executed). But if there are **D.** files, they will be taken care of by **D.** processing.

FILES

/usr/lib/uucp	directory with commands used by **uucleanup** internally
/var/spool/uucp	spool directory

SEE ALSO

uucp(1C), **uux**(1C), **cron**(8)

NAME

uucpd – UUCP server

SYNOPSIS

/usr/etc/in.uucpd

AVAILABILITY

This command is available with the *uucp* software installation option. Refer to *Installing SunOS 4.1* for information on how to install optional software.

DESCRIPTION

uucpd is the server for supporting UUCP connections over networks.

uucpd is invoked by **inetd**(8C) when a UUCP connection is established (that is, a connection to the port indicated in the "uucp" service specification; see **services**(5)), and executes the following protocol:

1) The server prompts with **login:**. The **uucico**(8C) process at the other end must supply a username.

2) Unless the username refers to an account without a password, the server then prompts with **Password:**. The **uucico** process at the other end must supply the password for that account.

If the username is not valid or is valid but refers to an account that does not have **/usr/lib/uucp/uucico** as its login shell, or if the password is not the correct password for that account, the connection is dropped. Otherwise, **uucico** is run, with the user ID, group ID, group set, and home directory for that account, with the environment variables **USER** and **LOGNAME** set to the specified username, and with a **–u** flag specifying the username. Entries are made in **/var/adm/wtmp** and **/var/adm/lastlog** for the username.

FILES

/var/adm/wtmp	accounting
/var/adm/lastlog	time of last login

SEE ALSO

services(5), **inetd**(8C), **uucico**(8C)

DIAGNOSTICS

All diagnostic messages are returned on the connection, after which the connection is closed.

user read
: An error occurred while reading the username.

passwd read
: An error occurred while reading the password.

Login incorect.
: The username is invalid or refers to an account with a login shell other than **/usr/lib/uucp/uucico**, or the password is not the correct password for the account.

NAME

uusched – the scheduler for the UUCP file transport program

SYNOPSIS

/usr/lib/uucp/uusched [**–u** *debug_level*] [**–x** *debug_level*]

AVAILABILITY

This command is available with the *uucp* software installation option. Refer to *Installing SunOS 4.1* for information on how to install optional software.

DESCRIPTION

uusched is the UUCP file transport scheduler. It is usually started by the daemon **uudemon.hour** that is started by **cron**(8) from an entry in the system **crontab** file:

39 * * * * /bin/su uucp –c "/usr/lib/uucp/uudemon.hour > /dev/null"

OPTIONS

–u *debug_level*

Pass *debug_level* as '**–x** *debug_level*' to any invocations of **uucico**(8C) started by **uusched**.

–x *debug_level*

Produce debugging output on the standard output. *debug_level* is a number between 0 and 9; higher numbers give more detailed information. 5, 7, and 9 are good numbers to try; they give increasing amounts of detail.

FILES

/etc/uucp/Systems
/etc/uucp/Permissions
/etc/uucp/Devices
/var/spool/uucp/*
/var/spool/locks/LCK*
/var/spool/uucppublic/*

SEE ALSO

uucp(1C), **uustat**(1C), **uux**(1C), **cron**(8), **uucico**(8C)

NAME

uuxqt – execute remote command requests

SYNOPSIS

/usr/lib/uucp/uuxqt [**–x** *debug_level*]

DESCRIPTION

uuxqt is the program that executes remote job requests from remote systems generated by the use of the **uux**(1C) command. **mail**(1) uses **uux** for remote mail requests. **uuxqt** searches the spool directories looking for **X.** files. For each **X.** file, **uuxqt** checks to see if all the required data files are available and accessible, and file commands are permitted for the requesting system. The **Permissions** file is used to validate file accessibility and command execution permission.

OPTIONS

–x *debug_level*

Produce debugging output on the standard output. *debug_level* is a number between 0 and 9; higher numbers give more detailed information. 5, 7, and 9 are good numbers to try; they give increasing amounts of detail.

ENVIRONMENT

There are two environment variables that are set before the **uuxqt** command is executed:

UU_MACHINE	Machine that sent the job (the previous one).
UU_USER	User that sent the job.

These can be used in writing commands that remote systems can execute to provide information, auditing, or restrictions.

FILES

/etc/uucp/Permissions
/etc/uucp/Maxuuxqts
/var/spool/uucp/*
/var/spool/locks/LCK*

SEE ALSO

mail(1), **uucp**(1C), **uustat**(1C), **uux**(1C), **uucico**(8C)

NAME

vipw – edit the password file

SYNOPSIS

/usr/etc/vipw

DESCRIPTION

vipw edits the password file while setting the appropriate locks, and does any necessary processing after the password file is unlocked. If the password file is already being edited, then you will be told to try again later. The vi(1) editor will be used unless the environment variable **VISUAL** or **EDITOR** indicates an alternate editor.

vipw performs a number of consistency checks on the password entry for root, and will not allow a password file with a "mangled" root entry to be installed. It also checks the **/etc/shells** file to verify the login shell for root.

FILES

/etc/ptmp
/etc/shells

SEE ALSO

passwd(1), **vi**(1), **passwd**(5), **adduser**(8)

NAME

vmstat – report virtual memory statistics

SYNOPSIS

vmstat [**–cfisS**] [*interval* [*count*]]

DESCRIPTION

vmstat delves into the system and normally reports certain statistics kept about process, virtual memory, disk, trap and CPU activity.

Without options, **vmstat** displays a one-line summary of the virtual memory activity since the system has been booted. If *interval* is specified, **vmstat** summarizes activity over the last *interval* seconds. If a *count* is given, the statistics are repeated *count* times.

For example, the following command displays a summary of what the system is doing every five seconds. This is a good choice of printing interval since this is how often some of the statistics are sampled in the system.

```
example% vmstat 5
 procs   memory              page    faults
 r b w  avm  fre  re at  pi  po  fr  de  sr x0 x1 x2 x3  in  sy  cs us sy id
 2 0 0  918  286  0 0   0   0   0   0   0  1  0  0  0   4  12   5  3  5 91
 1 0 0  846  254  0 0   0   0   0   0   0  6  0  1  0  42 153  31  7 40 54
 1 0 0  840  268  0 0   0   0   0   0   0  5  0  0  0  27 103  25  8 26 66
 1 0 0  620  312  0 0   0   0   0   0   0  6  0  0  0  26  76  25  6 27 67

CTRL-C
example%
```

The fields of **vmstat**'s display are:

procs Report the number of processes in each of the three following states:
- **r** in run queue
- **b** blocked for resources (i/o, paging, etc.)
- **w** runnable or short sleeper (< 20 secs) but swapped

memory Report on usage of virtual and real memory. Virtual memory is considered active if it belongs to processes which are running or have run in the last 20 seconds.
- **avm** number of active virtual Kbytes
- **fre** size of the free list in Kbytes

page Report information about page faults and paging activity. The information on each of the following activities is averaged each five seconds, and given in units per second.
- **re** page reclaims — but see the –S option for how this field is modified.
- **at** number of attaches — but see the –S option for how this field is modified.
- **pi** kilobytes per second paged in
- **po** kilobytes per second paged out
- **fr** kilobytes freed per second
- **de** anticipated short term memory shortfall in Kbytes
- **sr** pages scanned by clock algorithm, per-second

disk Report number of disk operations per second (this field is system dependent). For Sun systems, four slots are available for up to four drives: "x0" (or "s0" for SCSI disks), "x1", "x2", and "x3".

faults Report trap/interrupt rate averages per second over last 5 seconds.
- **in** (non clock) device interrupts per second
- **sy** system calls per second
- **cs** CPU context switch rate (switches/sec)

cpu Give a breakdown of percentage usage of CPU time.
- **us** user time for normal and low priority processes
- **sy** system time
- **id** CPU idle

OPTIONS

–c Report cache flushing statistics. By default, report the total number of each kind of cache flushed since boot time. The types are: user, context, region, segment, page, and partial-page.

–f Report on the number of **forks** and **vforks** since system startup and the number of pages of virtual memory involved in each kind of fork.

–i Report the number of interrupts per device. Autovectored interrupts (including the clock) are listed first.

–s Display the contents of the **sum** structure, giving the total number of several kinds of paging-related events which have occurred since boot.

–S Report on swapping rather than paging activity. This option will change two fields in **vmstat**'s ''paging'' display: rather than the ''re'' and ''at'' fields, **vmstat** will report ''si'' (swap-ins), and ''so'' (swap-outs).

FILES

/dev/kmem
/vmunix

BUGS

If more than one autovectored device has the same name, interrupts are counted for all like-named devices regardless of unit number. Such devices are listed with a unit number of '**?**'.

NAME

ypbatchupd – NIS batch update daemon

SYNOPSIS

/usr/etc/rpc.ypbatchupd

AVAILABILITY

Available only on Sun 386i systems running a SunOS 4.0.*x* release or earlier. Not a SunOS 4.1 release feature.

DESCRIPTION

ypbatchupd(8C) is the remote service used by **snap**(1) and **logintool**(8) to update the Network Interface Service (NIS) database on the master server, and to push all modified NIS maps to NIS servers. It is normally started by **/etc/rc.local**.

SEE ALSO

snap(1), **logintool**(8), **rc**(8)

NOTES

The Network Information Service (NIS) was formerly known as Sun Yellow Pages (YP). The functionality of the two remains the same; only the name has changed. The name Yellow Pages is a registered trademark in the United Kingdom of British Telecommunications plc, and may not be used without permission.

NAME

ypinit – build and install NIS database

SYNOPSIS

/usr/etc/yp/ypinit –m

/usr/etc/yp/ypinit –s *master_name*

DESCRIPTION

ypinit sets up a Network Interface Service (NIS) database on an NIS server. It can be used to set up a master or a slave server. You must be the super-user to run it. It asks a few, self-explanatory questions, and reports success or failure to the terminal.

It sets up a master server using the simple model in which that server is master to all maps in the data base. This is the way to bootstrap the NIS system; later if you want you can change the association of maps to masters.

Note: If there are both 3.x and 4.x NIS servers running in the network, the 4.x server should be configured as the master.

All databases are built from scratch, either from information available to the program at runtime, or from the ASCII data base files in **/etc**. These files are listed below under FILES. All such files should be in their "traditional" form, rather than the abbreviated form used on client machines.

An NIS database on a slave server is set up by copying an existing database from a running server. The *master_name* argument should be the hostname of an NIS server (either the master server for all the maps, or a server on which the data base is up-to-date and stable).

Read **ypfiles**(5) and **ypserv**(8) for an overview of the NIS service.

OPTIONS

–m Indicate that the local host is to be the NIS master.

–s Set up a slave database.

FILES

/etc/passwd
/etc/group
/etc/hosts
/etc/networks
/etc/services
/etc/protocols
/etc/ethers

SEE ALSO

ypfiles(5), **makedbm**(8), **ypmake**(8), **yppush**(8), **ypserv**(8), **ypxfr**(8)

NOTES

The Network Information Service (NIS) was formerly known as Sun Yellow Pages (YP). The functionality of the two remains the same; only the name has changed. The name Yellow Pages is a registered trademark in the United Kingdom of British Telecommunications plc, and may not be used without permission.

NAME

ypmake – rebuild NIS database

SYNOPSIS

cd /var/yp ; make [*map*]

DESCRIPTION

The file called **Makefile** in **/var/yp** is used by **make**(1) to build the Network Interface Service (NIS) database. With no arguments, **make** creates **dbm** databases for any NIS maps that are out-of-date, and then executes **yppush**(8) to notify slave databases that there has been a change.

If you supply a *map* on the command line, **make** will update that map only. Typing **make passwd** will create and **yppush** the password database (assuming it is out of date). Likewise, **make hosts** and **make networks** will create and **yppush** the host and network files, **/etc/hosts** and **/etc/networks**.

There are three special variables used by **make**: DIR, which gives the directory of the source files; NOPUSH, which when non-null inhibits doing a **yppush** of the new database files; and DOM, used to construct a domain other than the master's default domain. The default for DIR is **/etc**, and the default for NOPUSH is the null string.

Refer to **ypfiles**(5) and **ypserv**(8) for an overview of the NIS service.

FILES

/var/yp
/etc/hosts
/etc/networks

SEE ALSO

make(1), **ypfiles**(5), **makedbm**(8), **yppush**(8), **ypserv**(8)

NOTES

The Network Information Service (NIS) was formerly known as Sun Yellow Pages (YP). The functionality of the two remains the same; only the name has changed. The name Yellow Pages is a registered trademark in the United Kingdom of British Telecommunications plc, and may not be used without permission.

NAME

yppasswdd, rpc.yppasswdd – server for modifying NIS password file

SYNOPSIS

/usr/etc/rpc.yppasswdd *filename* [*adjunct_file*] [**–nogecos**] [**–noshell**] [**–nopw**]
[**–m** *argument1 argument2 ...*]

AVAILABILITY

This program is available with the *Networking* software installation option. Refer to *Installing SunOS 4.1* for information on how to install optional software.

DESCRIPTION

yppasswdd is a server that handles password change requests from **yppasswd**(1). Unless an *adjunct_file* is specified, it changes a password entry in *filename*, which is assumed to be in the format of **passwd**(5). *filename* is the password file that provides the basis for the *passwd.byname* and *passwd.byuid* maps. This should not be confused with the servers */etc/passwd* file which controls access to the server. In particular this file should not contain an entry for the super user.

If an *adjunct_file* is specified or **/etc/security/passwd.adjunct** exists, this file will be changed instead of the *filename*. An entry in *filename* or *adjunct_file* will only be changed if the password presented by **yppasswd**(1) matches the encrypted password of that entry.

If the **–noshell** **–nogecos** or **–nopw** options are given then these fields may not be changed remotely using **chfn**, **chsh**, or **passwd**(1).

If the **–m** option is given, then after *filename* or *adjunct_file* is modified, a **make**(1) will be performed in */var/yp*. Any arguments following the flag will be passed to *make*.

This server is not run by default, nor can it be started up from **inetd**(8C). If it is desired to enable remote password updating for the Network Interface Service (NIS), then an entry for **yppasswdd** should be put in the **/etc/rc** file of the host serving as the master for the NIS **passwd** file.

EXAMPLE

If the NIS password file is stored as **/var/yp/passwd**, then to have password changes propagated immediately, the server should be invoked as

/usr/etc/rpc.yppasswdd /var/yp/passwd –m passwd DIR=/var/yp

FILES

/var/yp/Makefile
/etc/security/passwd.adjunct
/etc/rc

SEE ALSO

make(1), **yppasswd**(1), **passwd**(1), **passwd**(5), **passwd.adjunct**(5), **ypfiles**(5), **inetd**(8C), **ypmake**(8)

NOTES

The password file specified to *rpc.yppasswdd* may not be a link.

The Network Information Service (NIS) was formerly known as Sun Yellow Pages (YP). The functionality of the two remains the same; only the name has changed. The name Yellow Pages is a registered trademark in the United Kingdom of British Telecommunications plc, and may not be used without permission.

NAME

yppoll – version of NIS map at NIS server

SYNOPSIS

/usr/etc/yp/yppoll [**–h** *host*] [**–d** *domain*] *mapname*

DESCRIPTION

yppoll asks a **ypserv**(8) process what the order number is, and which host is the Network Interface Service (NIS) master server for the named map. If the server is a v.1 NIS protocol server, **yppoll** uses the older protocol to communicate with it. In this case, it also uses the older diagnostic messages in case of failure.

OPTIONS

–h *host* Ask the **ypserv** process at *host* about the map parameters. If *host* is not specified, the NIS server for the local host is used. That is, the default host is the one returned by **ypwhich**(8).

–d *domain*
Use *domain* instead of the default domain.

SEE ALSO

ypfiles(5), **ypserv**(8), **ypwhich**(8)

NOTES

The Network Information Service (NIS) was formerly known as Sun Yellow Pages (YP). The functionality of the two remains the same; only the name has changed. The name Yellow Pages is a registered trademark in the United Kingdom of British Telecommunications plc, and may not be used without permission.

NAME

yppush – force propagation of changed NIS map

SYNOPSIS

/usr/etc/yp/yppush [**–v**] [**–d** *domain*] *mapname*

DESCRIPTION

yppush copies a new version of a Network Interface Service (NIS) map from the master NIS server to the slave NIS servers. It is normally run only on the master NIS server by the **Makefile** in **/var/yp** after the master databases are changed. It first constructs a list of NIS server hosts by reading the NIS map **ypservers** within the *domain*. Keys within the map **ypservers** are the ASCII names of the machines on which the NIS servers run.

A "transfer map" request is sent to the NIS server at each host, along with the information needed by the transfer agent (the program which actually moves the map) to call back the **yppush** . When the attempt has completed (successfully or not), and the transfer agent has sent **yppush** a status message, the results may be printed to stdout. Messages are also printed when a transfer is not possible; for instance when the request message is undeliverable, or when the timeout period on responses has expired.

Refer to **ypfiles**(5) and **ypserv**(8) for an overview of the NIS service.

OPTIONS

–d *domain*

Specify a *domain*.

–v Verbose. This prints messages when each server is called, and for each response. If this flag is omitted, only error messages are printed.

FILES

/var/yp/*domain***/ypservers.{***dir***,***pag***}**
/var/yp

SEE ALSO

ypfiles(5), **ypserv**(8), **ypxfr**(8)

NIS protocol specification

BUGS

In the current implementation (version 2 NIS protocol), the transfer agent is **ypxfr**(8), which is started by the **ypserv** program. If **yppush** detects that it is speaking to a version 1 NIS protocol server, it uses the older protocol, sending a version 1 YPPROC_GET request and issues a message to that effect. Unfortunately, there is no way of knowing if or when the map transfer is performed for version 1 servers. **yppush** prints a message saying that an "old-style" message has been sent. The system administrator should later check to see that the transfer has actually taken place.

NOTES

The Network Information Service (NIS) was formerly known as Sun Yellow Pages (YP). The functionality of the two remains the same; only the name has changed. The name Yellow Pages is a registered trademark in the United Kingdom of British Telecommunications plc, and may not be used without permission.

NAME

ypserv, ypbind, ypxfrd – NIS server and binder processes

SYNOPSIS

/usr/etc/ypserv [**–d**]

/usr/etc/ypbind [–s] [–ypset | –ypsetme]

ypxfrd [**–x**]

AVAILABILITY

This program is available with the *Networking* software installation option. Refer to *Installing SunOS 4.1* for information on how to install optional software.

DESCRIPTION

The Network Interface Service (NIS) provides a simple network lookup service consisting of databases and processes. The databases are **dbm**(3X) files in a directory tree rooted at **/var/yp**. These files are described in **ypfiles**(5). The processes are **/usr/etc/ypserv**, the NIS database lookup server, and **/usr/etc/ypbind**, the NIS binder. The programmatic interface to the NIS service is described in **ypclnt**(3N). Administrative tools are described in **yppush**(8), **ypxfr**(8), **yppoll**(8), **ypwhich**(8), and **ypset**(8). Tools to see the contents of NIS maps are described in **ypcat**(1), and **ypmatch**(1). Database generation and maintenance tools are described in **ypinit**(8), **ypmake**(8), and **makedbm**(8).

Both **ypserv** and **ypbind** are daemon processes typically activated at system startup time from **/etc/rc.local**. **ypserv** runs only on NIS server machines with a complete NIS database. **ypbind** runs on all machines using the NIS services, both NIS servers and clients.

ypxfrd transfers entire NIS maps in an efficient manner. For systems that use this daemon, map transfers will be 10 to 100 times faster, depending on the map. To use this daemon, **ypxfrd** should be run on a server running SunOS release 4.1. **ypxfr** will attempt to use **ypxfrd** first, if that fails, it will print a warning and then use the older transfer method.

The **ypserv** daemon's primary function is to look up information in its local database of NIS maps. The operations performed by **ypserv** are defined for the implementor by the *YP Protocol Specification*, and for the programmer by the header file **rpcsvc/yp_prot.h**. Communication to and from **ypserv** is by means of RPC calls. Lookup functions are described in **ypclnt**(3N), and are supplied as C-callable functions in the C library. There are four lookup functions, all of which are performed on a specified map within some NIS domain: **match**, **get_first**, **get_next**, and **get_all**. The **match** operation takes a key, and returns the associated value. The **get_first** operation returns the first key-value pair from the map, and **get_next** can be used to enumerate the remainder. **get_all** ships the entire map to the requester as the response to a single RPC request.

Two other functions supply information about the map, rather than map entries: **get_order_number**, and **get_master_name**. In fact, both order number and master name exist in the map as key-value pairs, but the server will not return either through the normal lookup functions. If you examine the map with **makedbm**(8), however, they will be visible. Other functions are used within the NIS service subsystem itself, and are not of general interest to NIS clients. They include **do_you_serve_this_domain?**, **transfer_map**, and **reinitialize_internal_state**.

The function of **ypbind** is to remember information that lets client processes on a single node communicate with some **ypserv** process. **ypbind** must run on every machine which has NIS client processes; **ypserv** may or may not be running on the same node, but must be running somewhere on the network.

The information **ypbind** remembers is called a *binding* — the association of a domain name with the internet address of the NIS server, and the port on that host at which the **ypserv** process is listening for service requests. This information is cached in the directory **/var/yp/binding** using a filename of **domainname.version**.

The process of binding is driven by client requests. As a request for an unbound domain comes in, the **ypbind** process broadcasts on the net trying to find a **ypserv** process that serves maps within that domain. Since the binding is established by broadcasting, there must be at least one **ypserv** process on every net. If

the client is running in C2 secure mode, then **ypbind** will only accept bindings to servers where the **ypserv** process is running as root. Once a domain is bound by a particular **ypbind**, that same binding is given to every client process on the node. The **ypbind** process on the local node or a remote node may be queried for the binding of a particular domain by using the **ypwhich**(1) command.

Bindings and rebindings are handled transparently by the C library routines. If **ypbind** is unable to speak to the **ypserv** process it's bound to, it marks the domain as unbound, tells the client process that the domain is unbound, and tries to bind the domain once again. Requests received for an unbound domain will wait until the domain requested is bound. In general, a bound domain is marked as unbound when the node running **ypserv** crashes or gets overloaded. In such a case, **ypbind** will to bind any NIS server (typically one that is less-heavily loaded) available on the net.

ypbind also accepts requests to set its binding for a particular domain. The request is usually generated by the NIS subsystem itself. **ypset**(8) is a command to access the **set_domain** facility. It is for unsnarling messes. Note: the **set_domain** procedure only accepts requests from processes running as root.

OPTIONS

–d The NIS service should go to the DNS (Domain Name Service) for more host information.

–s Secure. When specified, only ypservers bound to a reserved port are used. This allows for a slight increase in security in completely controlled environments, where there are no computers operated by untrusted individuals. It offers no real increase in security.

–v Do not fork when **ypxfrd** is called multiple times.

–ypset **ypset**(8) may be used to change the binding. This option is very dangerous, and only should be used for debugging the network from a remote machine.

–ypsetme

ypset(8) may be issued from this machine, security is based on IP address checking, which can be defeated on network where untrusted individuals may inject packets. This option is not recommended.

FILES

If the file **/var/yp/ypserv.log** exists when **ypserv** starts up, log information will be written to this file when error conditions arise.

The file(s) **/var/yp/binding/domainname.version** will be created to speed up the binding process. These files cache the last successful binding created for the given domain, when a binding is requested these files are checked for validity and then used.
/var/yp
/usr/etc/ypbind

SEE ALSO

domainname(1), **ypcat**(1), **ypmatch**(1), **dbm**(3X), **ypclnt**(3N), **ypfiles**(5) **makedbm**(8), **ypmake**(8), **ypinit**(8), **yppoll**(8), **yppush**(8), **ypset**(8), **ypwhich**(8), **ypxfr**(8),

Network Programming
System and Network Administration

NOTES

Both **ypbind** and **ypserv** support multiple domains. The **ypserv** process determines the domains it serves by looking for directories of the same name in the directory **/var/yp**. It will reply to all broadcasts requesting yp service for that domain. Additionally, the **ypbind** process can maintain bindings to several domains and their servers, the default domain is however the one specified by the **domainname**(1) command at startup time.

The Network Information Service (NIS) was formerly known as Sun Yellow Pages (YP). The functionality of the two remains the same; only the name has changed. The name Yellow Pages is a registered trademark in the United Kingdom of British Telecommunications plc, and may not be used without permission.

NAME

ypset – point ypbind at a particular server

SYNOPSIS

/usr/etc/yp/ypset [**–V1**|**–V2**] [**–d** *domain*] [**–h** *host*] *server*

DESCRIPTION

ypset tells **ypbind** to get the Network Interface Service (NIS) for the specified *domain* from the **ypserv** process running on *server*. If *server* is down, or is not running **ypserv**, this is not discovered until an NIS client process tries to get a binding for the domain. At this point, the binding set by **ypset** is tested by **ypbind**. If the binding is invalid, **ypbind** attempts to rebind for the same domain.

ypset is useful for binding a client node which is not on a broadcast net, or is on a broadcast net which is not running an NIS server host. It also is useful for debugging NIS client applications, for instance where an NIS map only exists at a single NIS server host.

In cases where several hosts on the local net are supplying NIS services, it is possible for **ypbind** to rebind to another host even while you attempt to find out if the **ypset** operation succeeded. For example, you can type:

```
example% ypset host1
example% ypwhich
host2
```

which can be confusing. This is a function of the NIS service subsystem's attempt to load-balance among the available NIS servers, and occurs when *host1* does not respond to **ypbind** because it is not running **ypserv** (or is overloaded), and *host2*, running **ypserv**, gets the binding.

server indicates the NIS server to bind to, and can be specified as a name or an IP address. If specified as a name, **ypset** attempts to use NIS services to resolve the name to an IP address. This works only if the node has a current valid binding for the domain in question. In most cases, *server* should be specified as an IP address.

Refer to **ypfiles**(5) and **ypserv**(8) for an overview of the NIS service.

OPTIONS

–V1 Bind *server* for the (old) v.1 NIS protocol.

–V2 Bind *server* for the (current) v.2 NIS protocol.

If no version is supplied, **ypset**, first attempts to set the domain for the (current) v.2 protocol. If this attempt fails, **ypset**, then attempts to set the domain for the (old) v.1 protocol.

–h*host* Set ypbind's binding on *host*, instead of locally. *host* can be specified as a name or as an IP address.

–d*domain*
Use *domain* , instead of the default domain.

DIAGNOSTICS

Sorry, I couldn't send my rpc message to ypbind on host *name*
The user is not root, or **ypbind** was run without one of the **–ypset** flags. See **ypserv**(8) for explanations of the **–ypset** flags.

SEE ALSO

ypwhich(1), **ypfiles**(5), **ypserv**(8)

NOTES

The Network Information Service (NIS) was formerly known as Sun Yellow Pages (YP). The functionality of the two remains the same; only the name has changed. The name Yellow Pages is a registered trademark in the United Kingdom of British Telecommunications plc, and may not be used without permission.

NAME

ypsync – collect most up-to-date NIS maps

SYNOPSIS

/usr/etc/yp/ypsync [–r] [–u]

AVAILABILITY

Available only on Sun 386i systems running a SunOS 4.0.*x* release or earlier. Not a SunOS 4.1 release feature.

DESCRIPTION

ypsync gathers current Network Information Service (NIS) maps to the local NIS server. When invoked with no arguments, it polls all the NIS servers listed in the **/etc/ypservers** NIS map for the maps they serve, and the order of those maps. If there are any new maps that the local server does not have, or if there are maps that are more current than the local server's copy, it excutes **ypxfr**(8) to transfer those maps to the local server.

ypsync eliminates the need for **cron**(8) jobs to ensure that NIS map updates are eventually transmitted to all NIS servers, and supports different NIS maps having different masters. It is invoked periodically by **ypserv**(8).

OPTIONS

–r When invoked with the **–r** flag, **ypsync** re-creates the local **/var/yp** directory and databases if needed. This facility is used when upgrading servers, since they can automatically retrieve NIS maps without needing manual intervention. The NIS master of the **ypservers** map can also designate new servers, which would automatically pick up their new maps on reboot.

–u When invoked with the **–u** flag, **ypsync** updates the list of NIS servers on the master of the **ypservers** NIS map to include the local system if it does not already, and then get copies of all the NIS databases. A user invoking **ypsync –u** may not be root, and must have the *networks privilege in the* NIS **group** map.

FILES

/var/yp/YP.*domainname*

SEE ALSO

ypupdate(3), **ypserv**(8), **ypxfr**(8)

Sun386i Advanced Administration

System and Network Administration

NOTES

The Network Information Service (NIS) was formerly known as Sun Yellow Pages (YP). The functionality of the two remains the same; only the name has changed. The name Yellow Pages is a registered trademark in the United Kingdom of British Telecommunications plc, and may not be used without permission.

NAME

ypupdated, rpc.ypupdated – server for changing NIS information

SYNOPSIS

rpc.ypupdated [**–is**]

DESCRIPTION

ypupdated is a daemon that updates information in the Network Interface Service (NIS), normally started up by **inetd**(8C). **ypupdated** consults the file **updaters**(5) in the directory **/var/yp** to determine which NIS maps should be updated and how to change them.

By default, the daemon requires the most secure method of authentication available to it, either DES (secure) or UNIX (insecure).

OPTIONS

–i Accept RPC calls with the insecure AUTH_UNIX credentials. This allows programmatic updating of the NIS maps in all networks.

–s Accept only calls authenticated using the secure RPC mechanism (AUTH_DES authentication). This disables programmatic updating of the NIS maps unless the network supports these calls.

FILES

/var/yp/updaters

SEE ALSO

updaters(5), **inetd**(8C), **keyserv**(8C)

System and Network Administration
Network Programming

NOTES

The Network Information Service (NIS) was formerly known as Sun Yellow Pages (YP). The functionality of the two remains the same; only the name has changed. The name Yellow Pages is a registered trademark in the United Kingdom of British Telecommunications plc, and may not be used without permission.

NAME

ypxfr – transfer NIS map from NIS server to here

SYNOPSIS

/usr/etc/yp/ypxfr [**–b**] [**–c**] [**–f**] [**–d** *domain*] [**–h** *host*] [**–s** *domain*] [**–C** *tid prog ipadd port*] *mapname*

DESCRIPTION

ypxfr moves a Network Interface Service (NIS) map in the default domain for the local host to the local host by making use of normal NIS services. It creates a temporary map in the directory **/var/yp/***domain* (this directory must already exist; *domain* is the default domain for the local host), fills it by enumerating the map's entries, fetches the map parameters (master and order number), and loads them. It then deletes any old versions of the map and moves the temporary map to the real *mapname*.

If run interactively, **ypxfr** writes its output to the terminal. However, if it is invoked without a controlling terminal, and if the log file **/var/yp/ypxfr.log** exists, it will append all its output to that file. Since **ypxfr** is most often run from the super-user's **crontab** file, or by **ypserv**, you can use the log file to retain a record of what was attempted, and what the results were.

If **issecure**(3) is TRUE, **ypxfr** requires that **ypserv** on the *host* be running as root. If the map being transferred is a secure map, **ypxfr** sets the permissions on the map to 0600.

For consistency between servers, **ypxfr** should be run periodically for every map in the NIS data base. Different maps change at different rates: the *services.byname* map may not change for months at a time, for instance, and may therefore be checked only once a day (in the wee hours). You may know that *mail.aliases* or *hosts.byname* changes several times per day. In such a case, you may want to check hourly for updates. A **crontab**(5) entry can be used to perform periodic updates automatically. Rather than having a separate **crontab** entry for each map, you can group commands to update several maps in a shell script. Examples (mnemonically named) are in **/usr/etc/yp**: **ypxfr_1perday**, **ypxfr_2perday**, and **ypxfr_1perhour**. They can serve as reasonable first cuts.

Refer to **ypfiles**(5) and **ypserv**(8) for an overview of the NIS service.

OPTIONS

–b Preserve the resolver flag in the map during the transfer.

–c Do not send a "Clear current map" request to the local **ypserv** process. Use this flag if **ypserv** is not running locally at the time you are running **ypxfr**. Otherwise, **ypxfr** will complain that it cannot talk to the local **ypserv**, and the transfer will fail.

–f Force the transfer to occur even if the version at the master is not more recent than the local version.

–d *domain*
Specify a domain other than the default domain.

–h *host* Get the map from *host*, regardless of what the map says the master is. If *host* is not specified, **ypxfr** asks the NIS service for the name of the master, and tries to get the map from there. *host* may be a name or an internet address in the form *a.b.c.d*.

–s *domain*
Specify a source domain from which to transfer a map that should be the same across domains (such as the *services.byname* map).

–C*tid prog ipadd port*
This option is **only** for use by **ypserv**. When **ypserv** invokes **ypxfr**, it specifies that **ypxfr** should call back a **yppush** process at the host with IP address *ipaddr*, registered as program number *prog*, listening on port *port*, and waiting for a response to transaction *tid*.

FILES

/var/yp/ypxfr.log log file
/usr/etc/yp/ypxfr_1perday
script to run one transfer per day, for use with **cron**(8)
/usr/etc/yp/ypxfr_2perday
script to run two transfers per day
/usr/etc/yp/ypxfr_1perhour
script for hourly transfers of volatile maps
/var/yp/*domain* NIS domain
/var/spool/cron/crontabs/root
Super-user's **crontab** file

SEE ALSO

issecure(3), **crontab**(5), **ypfiles**(5), **cron**(8), **ypserv**(8), **yppush**(8)

YP Protocol Specification, in *Network Programming*

NOTES

The Network Information Service (NIS) was formerly known as Sun Yellow Pages (YP). The functionality of the two remains the same; only the name has changed. The name Yellow Pages is a registered trademark in the United Kingdom of British Telecommunications plc, and may not be used without permission.

NAME

zdump – time zone dumper

SYNOPSIS

zdump [–v] [–c *cutoffyear*] [*zonename* ...]

DESCRIPTION

zdump prints the current time in each *zonename* named on the command line.

OPTIONS

–v For each *zonename* on the command line, print 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*
Cut off the verbose output near the start of the year *cutoffyear*.

FILES

/usr/share/lib/zoneinfo standard zone information directory

SEE ALSO

ctime(3V), **tzfile**(5), **zic**(8)

NAME

zic – time zone compiler

SYNOPSIS

zic [**–v**] [**–d** *directory*] [**–l** *localtime*] [*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 '–', the standard input is read.

Input lines are made up of fields. Fields are separated from one another by any number of white space characters. Leading and trailing white space on input lines is ignored. An '#' (unquoted sharp character) in the input introduces a comment which extends to the end of the line the sharp character appears on. White space characters and sharp characters 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 1:00 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–west 9:30 Aus CST 1987 Mar 15 2:00

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

–v Complain 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* Create time conversion information files in the directory **directory** rather than in the standard directory **/usr/share/lib/zoneinfo**.

–l *timezone* Use the time zone *timezone* as local time. **zic** will act as if the file contained a link line of the form

Link *timezone* **localtime**

FILES

/usr/share/lib/zoneinfo standard directory used for created files

SEE ALSO

time(1V), **ctime**(3V), **tzfile**(5), **zdump**(8)

NOTES

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.

Index

B

C

D

E

F

G

H

J

K

L

M

N

O

P

Q

R

T

U

V

W

X

Y

Z