FUJITSU COMPUTERS SIEMENS

SINIX/windows ONLINE Documentation

Reliant UNIX 5.44
System Administrator Guide

Edition September 1997

SINIX® Copyright © Siemens Nixdorf Informationssysteme AG 1990.
SINIX is the UNIX® System derivative of Siemens Nixdorf Informationssysteme AG.
Reliant® is a registered trademark of Siemens Pyramid Information Systems, Inc.
UNIX is a registered trademark in the United States and other countries, licensed exlusively through X/Open Company Limited.
Base: OSF/Motif®, Copyright © Open Software Foundation, Inc.
X Window System®, Copyright © Massachusetts Institute of Technology.
OSF/Motif is a registered trademark of Open Software Foundation, Inc.
X Window System is a registered trademark of Massachusetts Institute of Technology.
Copyright © Siemens Nixdorf Informationssysteme AG 1997.
All rights reserved. Delivery subject to availability; right of technical modifications reserved.
All hardware and software names used are trademarks of their respective manufacturers.
Siemens Nixdorf Informationssysteme AG

---

1 Preface

The Reliant UNIX operating system forms the link between the computer system and its users. It manages the complex web of components that make up a computer and allows the user to access these components in a straightforward manner.
The distinguishing features of Reliant UNIX are:

- Reliant UNIX is an interactive operating system, which means that there is direct communication between the user and the computer.
- Reliant UNIX is a multiuser operating system, which means that more than one user can work on the computer at the same time.
- Reliant UNIX is a multitasking operating system, which means that the computer can work on more than one task at a time.

1.1 Target group

This manual is intended for Reliant UNIX system administrators who already have experience of working with the Reliant UNIX operating system. It provides a fundamental survey of Reliant UNIX system administration on RM200, RM300, RM400 and RM600 systems. References to sources of further information are included at appropriate points in the body of the text and in the "Related publications" section at the back of the manual.

1.2 Requirements

On Reliant UNIX systems, the *root* login is used for most system administration work. Unless otherwise indicated, the actions described in this manual are assumed to be performed under the *root* login.
Note that the input and output shown in the examples are system-dependent and so will not necessarily be exactly the same in all details as on your system.

1.3 Changes since the last version of the manual

This edition supersedes the manual "SINIX V5.43, System Administrator's Guide", order number U23123-J-Z145-1-7600.
The documentation has been adjusted to the software version Reliant UNIX 5.44. The following changes have been made since the edition of May 1995:

- The command documentation has been adjusted to the current man pages.
- The special system administration features concerning Intel-based systems have been excluded.
- The description of SINIX/windows is based on the "Common Desktop Environment" (CDE).
- The section "Networking your computer" has been revised.
- The chapter "Troubleshooting" is no longer contained in this manual. For detailed information on error recovery refer to the manual "Error Diagnosis and Error Handling" "[31]".
- The description of hardware and accessories reflects the latest technology.
- The list of related publications was updated with titles of system-specific and further publications.

The structure of the manual has not changed. The "System Administrator's Guide" provides a broad overview of the duties of the system administrator. System-specific information has been excluded as far as possible. References to sources of further system-specific information are included at appropriate points.

1.4 Notational conventions

This manual adopts the following notational conventions:

Warning:

This symbol alerts you to potential causes of data loss or equipment damage.

Note:

This symbol draws your attention to important information and to references to related publications.

italics
: An *italic* font is used in the body of the text for commands, options, variables, file and path names, user input and operating system output.

`fixed-width`
: Operating system output in the examples is shown in a `fixed-width` font.

`fixed-width bold`
: User input in examples is shown in a **`fixed-width bold`** font. All lines of input in this font are terminated with the **`[RETURN]`** key. Consequently the **`[RETURN]`** key is not explicitly shown at the end of such lines.
Comments used to annotate input and output in examples are separated from the I/O by a hash sign **`#`**. The comment text is likewise shown in the **`fixed-width bold`** font.

`[Key symbols]`
: Keys are shown as they appear on the keyboard. If a letter specifically has to be entered in uppercase, the shift key is shown as well, e.g. **`[SHIFT]`** - **`[A]`** for A.
If two keys need to be pressed at the same time, they are joined together by a dash, as above.

If you can also perform a function described in the body of the text by using the sysadm menu interface, the description is followed by a table indicating the steps you need to take with sysadm. The table contains the following information: the names of the menu items which select the corresponding function, a brief description of the function, and the shell command (without options) on which the sysadm operation is based.
References to other places in this manual take the form either of a single page number or of a chapter or section heading and the number of the first page of the chapter or section.
References to other publications take the form of the name of the publication followed by square brackets enclosing the number allocated to it in the "Related publications" section at the back of the manual (e.g. "[9]").

2 Guided tour of the Reliant UNIX system

This chapter describes the characteristics of a Reliant UNIX system from a hardware point of view. It covers hard disk partitioning, file system organization and file system types, the directory structure and file type classification. Every system administrator should devote some time to getting to know these features of the operating system.

2.1 Hard disk partitioning

A hard disk on a Reliant UNIX system is not treated as a single entity. It is divided into a number of areas known as partitions. Under Reliant UNIX, every hard disk can be divided into up to 16 partitions (numbered 0 through 15).

Note:

A hard disk can also be split up into a number of areas each used by a different operating system. Areas of this type are often also referred to as "partitions". In this manual the term "partition" refers only to the subdivisions of a Reliant UNIX hard disk.

As the hardware is shipped with the operating system preloaded, the hard disk is already partitioned when you receive it. The partition layout will be similar to that shown in the table "Typical partition layout". You should not change the partition layout unless you reinstall the operating system. Then you can define the size and in some cases the file system type of individual partitions. For further information on installing the operating system refer to section "Installing the operating system" or to the Installation Guide for your system ("[2]", "[3]").
You can use the *autoconf -l* command to list the hard disk configuration data of your system. Note that the exact data shown in the following examples depends on the type of hard disk which is installed in the system.

```
# autoconf -l
System Configuration:
onbrd00
cpu00 ConfigReg=0x0180E48B PRId-Impl=4 PRId-Rev=5.0
cpu00 PICSz=16384 PDCSz=16384 PICLineSz=16 PDCLineSz=16
mem00 ram:0x2000000 SCMod=MIXED SCSize=131072 SCBSize=64
brd00 IDPROM:ser=02418 brdtype=5 ctype=2 mlyout=C clyout=D
ios0/scon00 IOS SCSI Host Adaptor NCRC710, single ended
ios0/stape003 MC12
ios0/sdisk000 MP81 -- MP81
ios0/sdisk005 OS02 (CD-ROM) -- OS02 (CD-ROM)
et00 i82596 LAN Rev. C [0:0:e4:3:9:72]
du00 SC2681 (Port A/B)
term/00 SC2681(Port A): console/mouse
term/01 SC2681(Port B): tty (modem control)
term/02 SC2681(Port B): DIAL IN (modem control)
term/03 SC2681(Port B): DIAL OUT (modem control)
term/04 SC2681(Port B): TELESERVICE (modem control)
du01 VL16C552 (Port 0/1)
term/05 VL16C552(Port 0): tty (modem control)
```

```
term/06 VL16C552(Port 1): tty (modem control)
eisa On-board EISA-BUS
fd01 i82077AA Floppy Disk Subsystem Controller
flp/fx3h 3.5" DS/HD Floppy Disk Drive
md/mdisk0 memory disk -- no memory allocated
#
#
```

The *dkpart -l special-file* command shows you how the hard disk is currently partitioned.

```
# dkpart -l /dev/ios0/rsdisk000s0
ios0/sdisk000 "MP81"     Geometry 2694-2:9:85 (cyls:heads:sectors)
Sector size = 512 bytes  1006.3 Total Mbytes (1MB = 1024 * 1024 bytes)
Partition Start Rule End Rule  First Cyl  Last Cyl  Cyl Count  Size (MB)
0         6          188          6       193        188       70.2
1         >p0        343        194       536        343      128.1
2         >p1        536        537      1072        536      200.2
3         >p2        456       1073      1528        456      170.3
4         >p3        =p2/2     1529      1796        268      100.1
5         >p4        >p7       1797      2690        894      333.9
6         3          >p3          0      1528       1529      571.1
7         0          $/3*3        0      2690       2691     1005.2
8         >p6        >p5       1529      2690       1162      434.0
9         0          0                * * *    Unused    * * *
10         0          6             0         5          6        2.2
11         0          764           0       763        764      285.4
12         >p11       >p3         764      1528        765      285.6
13         >p12       581        1529      2109        581      217.0
14         >p13       >p5        2110      2690        581      217.0
15         >p14       1          2691      2691          1        0.4
# Partition layout (not to scale):
|10-|-0--|---1----|------2------|-----3-----|--4---|-------5-------15
|---------------------6---------------------|----------8-----------|
|---------------------------------7---------------------------------|
|-----------11-----------|--------12--------|-----13-----|----14----|
#
```

Note:

For detailed descriptions of the *autoconf* and *dkpart* commands refer to the "System Administrator's Reference Manual" "[7]".

The following table shows how a hard disk is partitioned on a Reliant UNIX system:

Partition	**Contents**
Partition 0	Contains the / partition (*root*)
Partition 1	Contains the primary swap area for holding processes or parts of processes swapped out of memory
Partition 2	Contains the */opt* partition
Partition 3	Contains the */usr* partition, comprising commands and constant system data

Partition 4	Contains the */var* partition
Partition 5	Contains the */home* partition
Partition 6	Optional
Partition 7	References the entire hard disk (apart from partition 15)
Partition 8-9	Optional
Partition 10	Contains the SASH and the hard disk header information (volume header) (not on RM600-xx systems)
Partition 11-14	Optional
Partition 15	Reserved for a *statesave* partition

Table 1: Typical partition layout

Warning:

Partitions 6 through 9 and 11 through 14 must not be used on the system disk as these may overlap with the file systems of the operating system.

2.2 File systems

Directories and files under Reliant UNIX are grouped in logical, structured units known as file systems. A file system is a system of organization in which directories and files are managed. To simplify, we can view a file system as a part of the hard disk which the operating system treats as a coherent unit.
One reason for splitting up the hard disk into partitions, as described in the previous section, is to allow different file systems to be set up on the various partitions.

2.2.1 File systems and file system types

There are a number of methods of finding out which file systems are available on a computer. The output of the *mount* command provides information on the currently mounted file systems and their mount points. The following output is typical:

```
# mount
/ on /dev/root read/write/setuid on Mon Jun 16 08:23:20 1997
/proc on /proc read/write on Mon Jun 16 08:23:20 1997
/dev/fd on /dev/fd read/write on Mon Jun 16 08:23:20 1997
/opt on /dev/ios0/sdisk000s2 read/write/setuid/noquota on Mon Jun 16
08:23:23 1997
/usr on /dev/ios0/sdisk000s3 read/write/setuid/noquota on Mon Jun 16
08:23:24 1997
/var on /dev/ios0/sdisk000s4 read/write/setuid/noquota on Mon Jun 16
08:23:24 1997
/home on /dev/ios0/sdisk000s5 read/write/setuid/noquota on Mon Jun 16
08:23:25 1997
#
```

Each line shows a file system mount point and the block special file of the corresponding partition, followed by various access parameter values and the time when the file system was mounted.
A file system consists of at least one superblock, an inode table, and a storage/data area. The superblock maintains information about the entire file system, such as its size and the number of free blocks. The inode table, a list of the file system's information nodes (inodes for short), contains information for every file in the file system, such as its name, its path, its owner, its group affiliation, and the physical addresses of the data blocks which form the file.
Reliant UNIX supports a variety of file system types which exhibit considerable differences in terms of layout and characteristics. The main differences between the various file system types are related to the addressing mechanism and the block size.

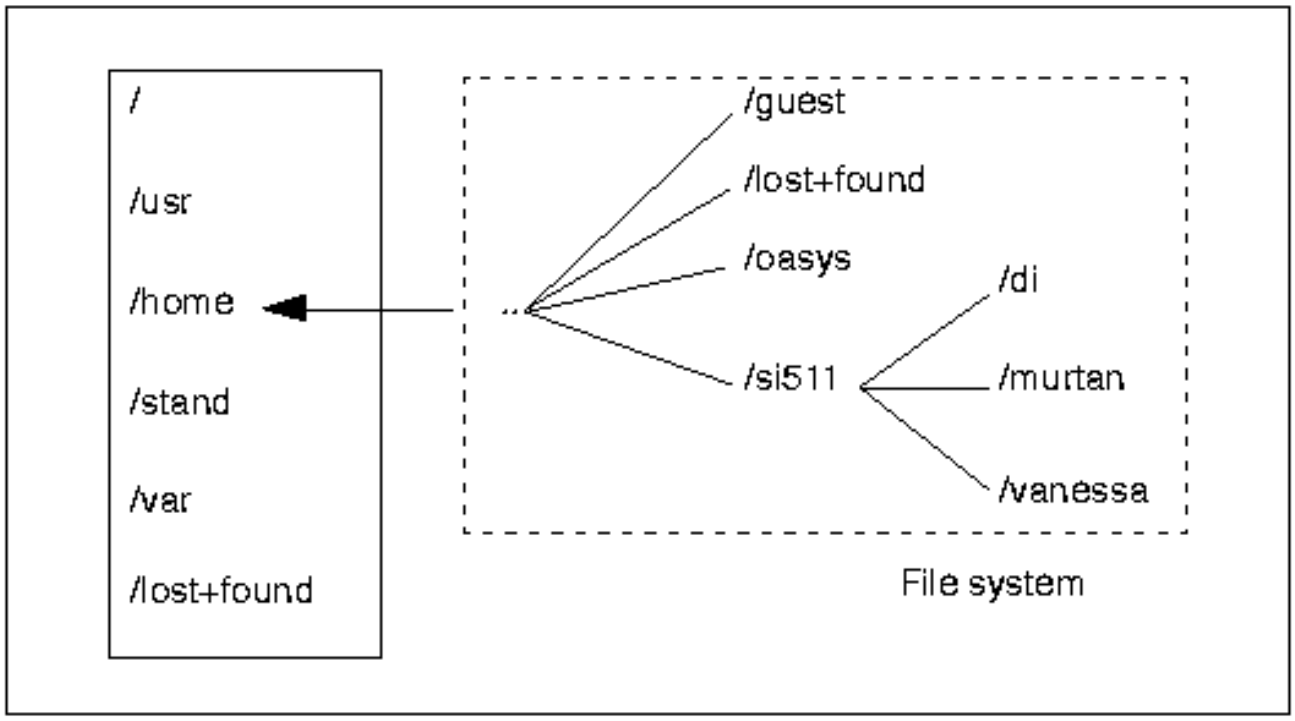

Figure 1: File system mounted on the /home directory

The following table summarizes the most common Reliant UNIX file system types:

File system type	**Explanation**
fifofs	Provides for common access to pipe files. See section "Named pipes".
hs	High Sierra/ISO 9660 file system for CD-ROM access. See section "File system types: hs".
nfs	Network file system; makes file systems available as distributed file systems which can be accessed throughout a network, regardless of operating system type. See section "File system types: nfs".
proc	Process file system; contains images of all active processes. See section "File system types: proc".
rr	Rockridge file system; as hs, but with fewer restrictions, for example the length of file names is not limited to 14 characters, special characters are allowed in file names, the depth of the file system is not restricted to eight levels.
specfs	Supplies a common code interface for all

	special files.
ufs	Unix file system; an implementation of the BSD fast file system. See section "File system types: ufs".
VxFS	Veritas high-availability file system. See section "File system types: VxFS (Veritas file system)"

Table 2: Reliant UNIX file system types

Reliant UNIX 5.4x simultaneously supports and manages multiple file system types. Thanks to the VFS (virtual file system) mechanism, the various file system types can all run in parallel on the same computer.
The term "virtual file system" denotes a file system architecture which features a precisely defined, modular interface between file systems and the Reliant UNIX kernel. This creates the conditions for concurrent use of a wide range of file system types under the operating system. The file systems may have widely differing properties and internal formats.

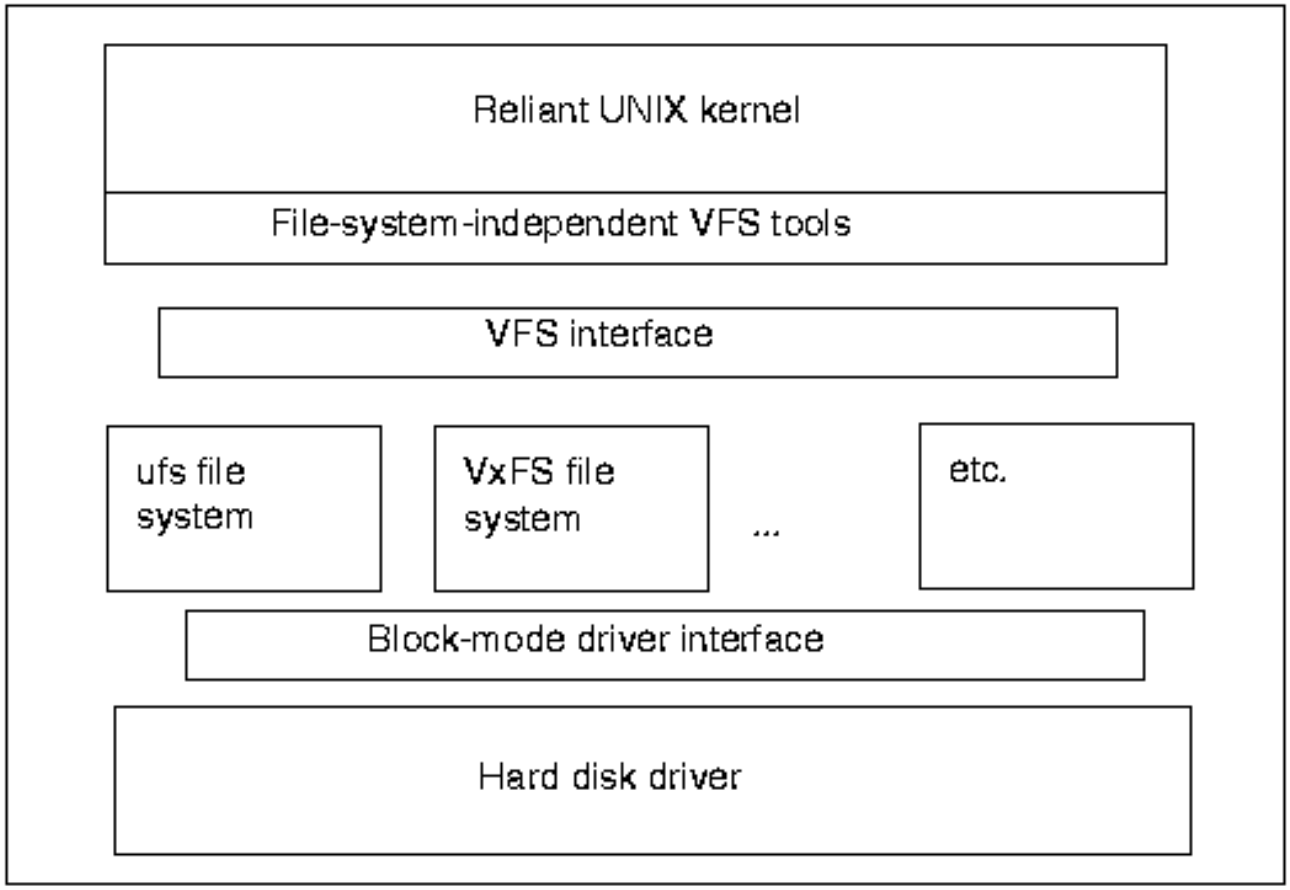

Figure 2: Modular file system integration in Reliant UNIX

The great advantage of the VFS architecture is the relative ease with which new file system types can be integrated, such as the hs file system type (High Sierra/ISO 9660 file system) for CD-ROMs or its successor, the Rockridge file system. The figure "Modular file system integration in Reliant UNIX" illustrates the modular integration of file system types in Reliant

UNIX.
On a Reliant UNIX system there is generally a file system installed on every partition. The exceptions to this rule are the partitions reserved for system information and the swap partition for swapping out executable programs.
There must also be at least one file system for each usable storage medium (hard disk, floppy disk, CD-ROM, MOD) which is attached to the computer to enable standard Reliant UNIX commands such as *cd* or *cp* to be applied to the storage media. In other words, each medium must be integrated in the overall Reliant UNIX file system and directory hierarchy. If the media are used for data backup, they are accessed using special-purpose backup commands (see section "Backup commands").
A file system cannot be made to span more than one media unit, two hard disks, for example. However, it is possible to circumvent this problem by defining virtual disks (for information on defining virtual disks refer to the "Virtual Disks" manual "[12]").
Each file system requires a mount point in the file system hierarchy. Mount points take the form of directories. When the operating system is booted, the various file systems are automatically mounted in a fixed sequence. This sequence is defined in the */etc/vfstab* file.

```
# cat /etc/vfstab
#
block                fsck                     mount     fs  fsck auto mount
# dev                  dev                      point    type pass mnt  opts
#
/proc                  -                        /proc    proc  -    -    rw
/dev/fd                -                        /dev/fd  fdfs  -    -    rw
/dev/ios0/sdisk000s0   /dev/ios0/rsdisk000s0    /        ufs   -    -    rw
/dev/ios0/sdisk000s1   /dev/ios0/rsdisk000s1    -        swap  -    -    rw
/dev/ios0/sdisk000s2   /dev/ios0/rsdisk000s2    /opt     ufs   0    yes  rw
/dev/ios0/sdisk000s3   /dev/ios0/rsdisk000s3    /usr     ufs   0    yes  rw
/dev/ios0/sdisk000s4   /dev/ios0/rsdisk000s4    /var     ufs   0    yes  rw
/dev/ios0/sdisk000s5   /dev/ios0/rsdisk000s5    /home    ufs   0    yes  rw
#
```

For further information on the */etc/vfstab* file refer to section "The /etc/vfstab file". File system mounting is discussed in section "Mounting file systems".
The */* file system (root) must always be present. It acts as the top-level file system and is the entry point for using the other file systems. Other file systems are mounted on directories in this top-level file system. It is essential for the system administrator to know that there are different file systems which need to be handled in different ways. File system handling is described in depth in section "Working with file systems".
You can build new file systems with the *mkfs* and *newfs* commands. Using the *mkfs -m* command you can find out the values used when a file system was built.

Warning:

Do not forget the *-m* option when using the *mkfs* command simply to find out how a file system has been configured, as otherwise the command will be used to create a new file system.

This example shows the values defined for partition 4, where the */var* file system is mounted.

```
# mkfs -m /dev/ios0/sdisk000s4
mkfs -F vxfs -o ninode=49152, ausize=34828, aufirst=258, aupad=0, bsize=1024,
logsize=256, indsize=8, iaddrlen=8, /dev/ios0/rsdisk000s4 409600
```

#

Note:

There is an in-depth description of the *mkfs* and *newfs* commands in the "System Administrator's Reference Manual" "[7]".

2.2.2 File system types: VxFS (Veritas file system)

The Veritas file system (VxFS), named after its vendor, offers advantages in terms of performance and availability over the ufs file system (see section "File system types: ufs"). For example, the system is back on-line more quickly after an operating system failure, and file systems can be reliably backed up even while the system is on-line (see chapter "High availability").
Among the chief features of the Veritas file system are:

- All file system information (storage management information, inodes and data blocks) is stored in a superblock.
- Division into allocation units (corresponding to the cylinder groups in a ufs file system). For the sake of security, a copy of the superblock is kept in each allocation unit.
- The Veritas file system groups contiguous data blocks to form extents. When allocating space for a file, Veritas can assign a complete extent, rather than just a single block as is the case with s5 and ufs file systems. That means that hard disk I/O to and from a file can operate on more than one block at a time.
- The Veritas file system uses a strategy known as "intent logging". This involves recording pending changes to the file system structure in a log file (the intent log). When a Veritas file system is recovering from a system failure, the VxFS *fsck* command reads the intent log and completes the operations which were active when the system failed.
- You can make backups while on-line by taking a snapshot of the file system. You can then mount this snapshot like an ordinary file system and process it using Reliant UNIX commands such as *tar* and *cpio*.
- You can defragment a file system and change its size while the file system is mounted.
- On systems with Large File System support (e.g. Reliant UNIX), the size of files or file systems is limited to 16 terabytes.
- There is no internal limit on the number of file systems mounted at one time or on the number of files accessed at the same time.

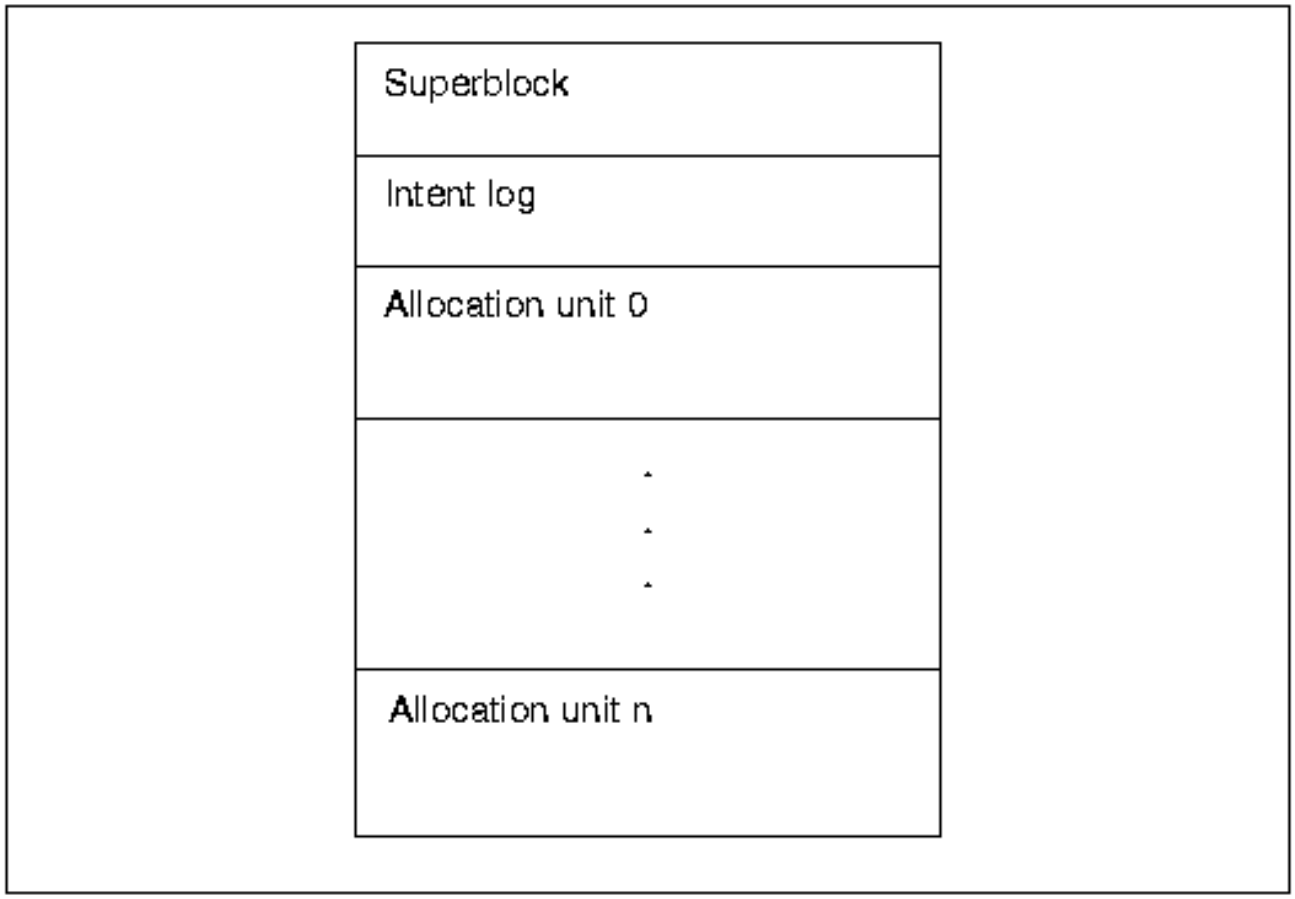

Figure 3: File system types: VxFS

Note:

For detailed information on the Veritas file system refer to the manual "Veritas File System (VxFS) V1.2, System Administrator's Guide" "[13]".

2.2.3 File system types: ufs

The ufs file system type (**u** nix **f**ile **s**ystem) can as a rule be used for all file systems which contain user data. Its features make it particularly suitable for systems where users primarily work with large files.
Among the particular features of the ufs file system type are:

- Division into cylinder groups.
- All file system information (storage management information, inodes and data blocks) is stored in a superblock.
- For the sake of security, a copy of the superblock is kept in each cylinder group.
- The standard data block size can be 4K, 8K or 16K. This provides for fast file access, particularly with large files.
- An optimization mechanism ensures that the blocks of a file are all stored in one cylinder group. This mechanism makes time-consuming reorganization of fragmented file systems unnecessary. The downside is that about ten percent of the space in the file

system is needed for the optimization mechanism.
- File names can be up to 255 characters long.
- The number of inodes is not restricted to 64,000.
- The file size is limited to 2 gigabytes, the file system size to 4 gigabytes.
- User quotas can be set up (see section "Defining user disk space quotas").

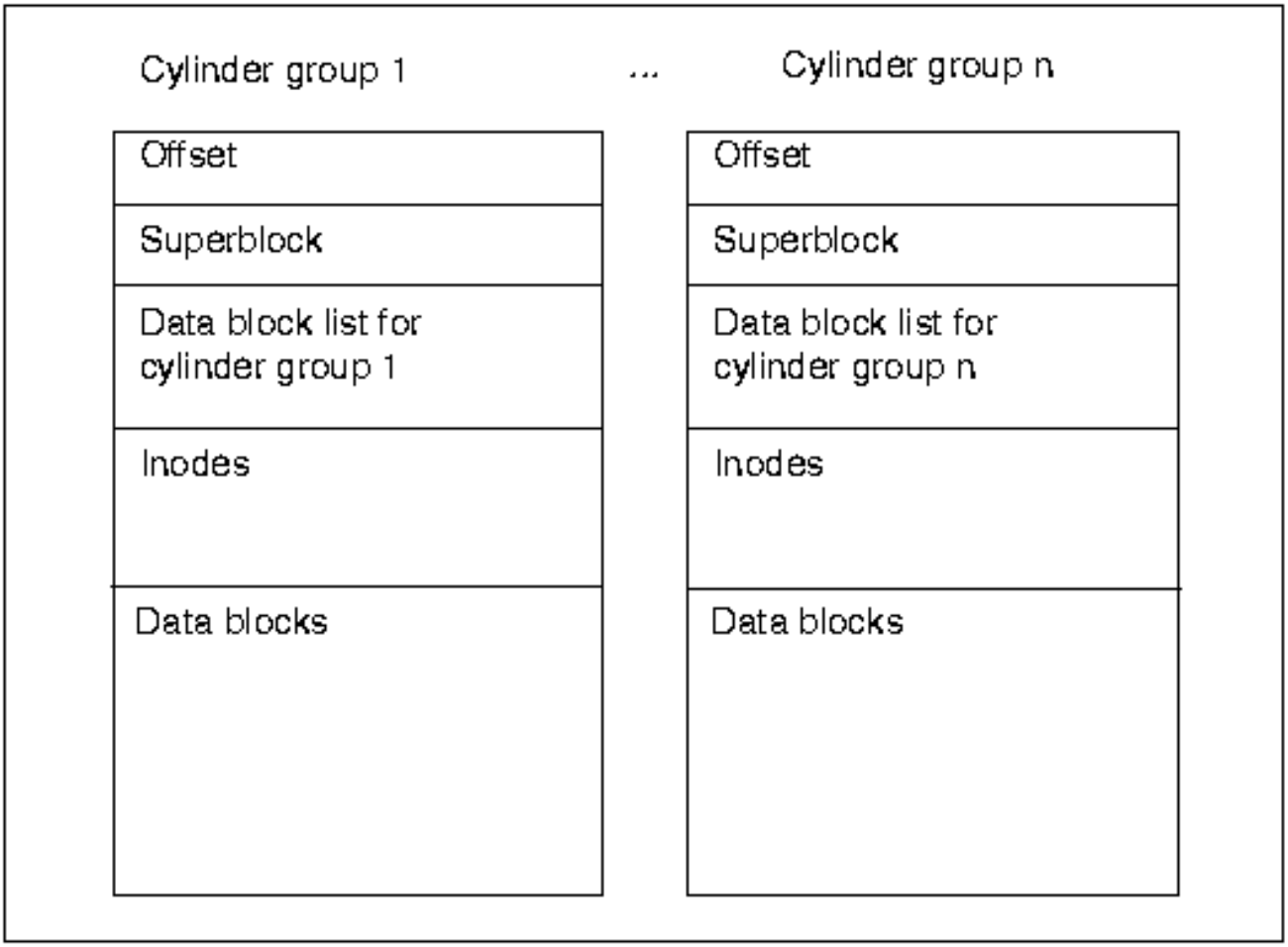

Figure 4: File system types: ufs

2.2.4 File system types: hs

A new file system type providing access to data on CD-ROMs has been integrated in Reliant UNIX. The **H** igh **S**ierra/ISO 9660 file system allows a CD-ROM to be treated as a read-only hard disk.
The file system is hierarchical: it contains directories which point to files or to other directories. Each directory contains "." and ".." entries.
In accordance with the High Sierra specification and the ISO 9660 standard, the names of directories and files are in the form *NAME;EXT;VERSION*.
VERSION consists of a string of digits, while *NAME* and *EXT* can be any string consisting of uppercase letters, digits and underscores; but to provide compatibility with DOS applications, an option has been added to the *mount* command (*mount -F hs -o dos <device_file> <mount_point>*) to handle name conversion based on the following rules:

- All uppercase letters are shifted to lowercase.
- Name searches do not distinguish between uppercase and lowercase.
- The version number is removed from the name.

2.2.5 File system types: nfs

The nfs file system type (**n** etwork **f**ile **s**ystem) supports distributed file access in networking environments encompassing dissimilar computer architectures and operating systems. This service allows a system to make local files and directories available for users of other computers to access with the aid of local access methods.
This is possible because the nfs file system utilizes an abstract file system model which is mapped onto the relevant local file system semantics. As a result, file system operations run exactly as they do on local file systems.

2.2.6 File system types: proc

For the dummy file system */proc* there is a special file system type: *proc* . The */proc* directory maintains images of all the processes currently running on the computer and is therefore known as a process file system. The *proc* file system type is referred to as a dummy file system because you cannot process, create or access *proc* file systems in the same way as other file systems. The contents of the file system can be listed with the *ls* command in the usual way; but the "file names" are actually process IDs, and the "file size" column indicates the space the process occupies in memory while it is executing.

```
# ls -l /proc
total 55648
-rw-------   1 root     root           0 Jul 18 09:32 00000
-rw-------   1 root     root      393216 Jul 18 09:32 00001
-rw-------   1 root     root           0 Jul 18 09:32 00002
-rw-------   1 root     root           0 Jul 18 09:32 00003
-rw-------   1 root     root           0 Jul 18 09:32 00004
-rw-------   1 root     root           0 Jul 18 09:32 00005
-rw-------   1 root     root           0 Jul 18 09:32 00006
...
-rw-------   1 root     root     1454080 Jul 18 09:32 00088
-rw-------   1 root     root       94208 Jul 18 09:32 00128
-rw-------   1 root     root      479232 Jul 18 09:32 00187
-rw-------   1 root     root      516096 Jul 18 09:32 00213
-rw-------   1 root     root      499712 Jul 18 09:32 00214
-rw-------   1 root     root      499712 Jul 18 09:32 00215
-rw-------   1 root     root      667648 Jul 18 09:32 00244
-rw-------   1 root     root      475136 Jul 18 09:32 00246
-rw-------   1 root     root     1204224 Jul 18 09:32 00250
-rw-------   1 root     root      573440 Jul 18 09:32 00294
...
#
```

2.3 Directory structure

When you log in as system administrator, you are placed at the topmost level of the directory hierarchy. To find your way around the directory structure you can use the *ls* command. In all there are several hundred directories, but naturally not all of them are discussed below. The command *find / -depth -type d -print | pg* lists the names of all the directories on your system.
There are various ways of viewing the structure of the top-level directories below the *root* directory on a Reliant UNIX system. One approach is shown in the following example:

```
# ls -l / | grep "^d"
drwxr-xr-x  2 bin    bin         512 Nov 22 10:30 bck
drwxr-xr-x  2 bin    bin         512 Nov 22 10:30 boot
drwxrwxrwx  2 root   root        512 Nov 22 10:30 cdrom
drwxrwxr-x  2 bin    bin         512 Nov 22 10:30 config
drwxr-xr-x 23 bin    bin        1536 Jan  4 14:26 dev
drwxrwxrwx  2 bin    bin         512 Nov 22 11:06 dgn
drwxrwxrwx 40 bin    bin        3072 Jan  5 09:43 etc
drwxrwxrwx  2 bin    bin         512 Nov 22 10:30 export
drwxr-xr-x 19 root   sys         512 Nov 25 15:44 home
dr-xr-xr-x  2 root   root      16384 Dec 18 05:18 lost+found
drwxr-xr-x  2 bin    bin         512 Nov 22 10:30 mnt
drwxr-xr-x 25 root   usrother    512 Nov 30 10:47 opt
drwxrwxr-x  2 root   root       8544 Jan  5 10:04 proc
drwxrwxrwt  4 bin    bin        2560 Nov 22 11:38 sbin
drwxr-xr-x  2 root   sys         512 Jan  4 14:26 stand
drwxr-xr-x  2 root   bin         512 Nov 22 10:57 tftpboot
drwxr-xr-x  2 bin    bin         512 Nov 22 10:30 tmp_mnt
drwxr-xr-x 28 bin    bin        1024 Nov 22 11:20 usr
drwxr-xr-x 25 bin    bin         512 Nov 29 17:05 var
#
```

As already mentioned in section "Hard disk partitioning", these directories are part of the *root* file system. Another way of listing the directories in the *root* file system is to use the *find* command:

```
# find / -depth -type d -mount -print
/lost+found
/etc/cron.d
...
/etc/default
/etc/dfs
/etc/fs/bfs
/etc/fs/events
/etc/fs/proc
/etc/fs/s5
/etc/fs/fdfs
/etc/fs/ufs
/etc/fs/hs
/etc/fs
...
/etc/rc.d
/etc/rc0.d
/etc/rc1.d
/etc/rc2.d
```

```
/etc/rc3.d
...
/etc/saf
/etc/shutdown.d
/etc/acct
...
/etc/lp
...
/etc/keytables
/etc/idrc.d
/etc/idsd.d
...
/etc/snmpd
/etc/sm
/etc
/cdrom
/bck
...
/dev
/dgn
/export
/mnt
/sbin
/stand
/tmp_mnt
/tftpboot
#
```

The following subsections briefly examine the principal Reliant UNIX directories.

2.3.1 /bck

The */bck* directory is used to mount a file system from which earlier backups can be restored.

2.3.2 /dev

The */dev* directory contains character and block special files together with subdirectories of special files which provide access to external devices such as hard disks, cartridge tape drives (streamers), floppy disk drives, printers and terminals (see section "Special files"). These files are associated with driver programs.

2.3.3 /dgn

The */dgn* directory stores diagnostic programs.

2.3.4 /etc

The */etc* directory contains system administration scripts, important machine-specific configuration files and login administration files.

2.3.5 /export

The */export* directory is an empty directory which is used to make files and directories available throughout a network. Under certain conditions, remote systems can mount this directory in their own directory hierarchy as an nfs file system (see the section "File systems and file system types").

2.3.6 /home

The */home* directory is where user accounts can be set up. From the user's viewpoint */home* is a directory, though it is actually the mount point for a self-contained file system.

2.3.7 /lost+found

The */lost+found* directory is by default set up for every file system. The *fsck* command uses it to store files and directories (see section "Checking and repairing file systems").

2.3.8 /mnt

The */mnt* directory is an empty directory which is often used as a temporary file system mount point (see also section "/tmp_mnt" and section "Mounting file systems").

2.3.9 /opt

The */opt* directory usually contains a VxFS file system (see section "File system types: VxFS (Veritas file system)"). The subdirectories of the file system may in turn contain executable programs, commands and libraries of additionally installed (optional) application software.

2.3.10 /proc

The */proc* directory is used to mount a *proc* file system (see section "File system types: proc"). It contains images of the processes currently in memory.

2.3.11 /sbin

The */sbin* directory contains executable programs, commands and shell scripts needed to boot the operating system and for system administration.

2.3.12 /shlib

The */shlib* directory is included to provide compatibility with UNIX System V Release 3. It contains libraries and help files.

2.3.13 /stand

The */stand* directory contains the kernel and system startup files.
Regular files can be copied into it, but no directories or special files can be created.

2.3.14 /tmp

/tmp is a symbolic link to */var/tmp* , where the operating system and software products store temporary files such as error messages and logs. These temporary files are deleted at

every shutdown or reboot.

2.3.15 /tmp_mnt

The */tmp_mnt* directory is used to mount file systems used on a temporary basis (also see section "/mnt" and section "Mounting file systems").

2.3.16 /usr

The */usr* directory is used as a mount point for the *usr* file system. This file system contains commands and constant system data.

2.3.17 /var

The */var* directory is the mount point for the *var* file system. The subdirectories of this file system contain variable data such as log files (for cron services and the Network Information Service NIS), accounting files (accounting service), electronic mail (*mail*), spool files, installation logs, backup logs and kernel messages.

2.3.18 Directory structure summary

The following table summarizes the directories and their contents:

Directory	Contents
/bck	Directory used as mount point for restoring backups
/dev	Special files
/dgn	Directory for diagnostic programs
/etc	Contains system and boot files
/export	Empty directory acting as mount point for file systems which can be accessed over a network
/home	Mount point for a file system containing user directories
/lost+found	Set up for each file system and needed for files restored by the *fsck* command
/mnt	Empty directory used as mount point for file systems
/opt	File system for optional software
/proc	Mount point for process file system
/sbin	Directory of system administration commands and scripts
/shlib	Directory providing compatibility with V5.3 systems
/stand	Directory containing the */stand/unix* kernel

/tmp	Symbolic link to */var/tmp*, where temporary files are stored.
/tmp_mnt	Directory for mounting temporarily used file systems
/usr	Directory acting as mount point for a file system containing user commands and constant system data
/var	Directory acting as mount point for a file system containing variable files such as spool, log and accounting files

Table 3: Directories under root (/)

2.4 Files

In Reliant UNIX there are no rules governing how files are structured. A file is simply a sequence of coded characters. No information is stored in a file about the file itself. Reliant UNIX does however distinguish between different file types, such as regular files, directories, symbolic links, named pipes and (device) special files.

2.4.1 Regular files

A file is a data store in which data can be saved and held accessible. A file may contain information such as the text of a report or the code of a program.
Below we discuss just two aspects of regular files, both relating to access permissions: the *umask* command and the t bit (sticky bit) for executable programs.
As system administrator you must use the *umask 066* command to protect your files against unauthorized reading, writing and deletion. Even nonprivileged users can add this command to their *$HOME/.profile* files to ensure that all new files are accessible only to the owner. *umask 066* is the default setting for the system administration accounts *root* and *sysadm*.
If the t bit is set on an executable program, the program is not removed from memory after it has been executed. That means that it does not need to be reloaded the next time it is called and is therefore available faster.

Note:

For a description of the *umask* command refer to the manual "Commands" "[5]". Access permissions are discussed in the "User's Guide" "[6]".

2.4.2 Directories

A directory is a type of file which a user can create or delete but cannot write to in the same way as a regular file. The entries in a directory file are links to (i.e. the names of) the files and other directories which the directory contains.
You can verify this by using a command which is not normally applied to directories. The output of the command *cat directory-name* admittedly includes much unreadable garbage, but the entries for files together with their inodes are clearly identifiable.
One special feature shared by directories and executable programs is the t bit (or sticky bit). Files in directories with the t bit set can be deleted only by the file owner. For a full explanation of the t bit refer to the description of the *chmod* command in the manual "Commands" "[5]" (also see section "Regular files").

```
# cd $HOME
# mkdir TEST
# ls -ld TEST
drwx--x--x 2 root root 512 Mar 17 10:14 TEST
# chmod 777 TEST
# ls -ld TEST
drwxrwxrwx 2 root root 512 Mar 17 10:14 TEST
# chmod +t TEST
# ls -ld TEST
drwxrwxrwt 2 root root 512 Mar 17 10:14 TEST
#
```

2.4.3 Symbolic links

A symbolic link is a file containing the path name of another file or of a directory. When a symbolic link is accessed, the kernel converts this to an access to the referenced file or directory. This mechanism also operates across file system boundaries. This even makes it possible to group directory structures and files to form logical directory trees which transcend machine boundaries.
Reliant UNIX 5.4x uses the symbolic link mechanism to avoid incompatibilities with earlier versions. One use you can make of it as system administrator is to install the man pages retroactively on another file system. The man pages are by default stored under the */usr/share/man* directory. The following is an example of installing them under */home/man*.

```
# cd /usr/share/man
# ln -s /home/man/cat1 cat1
...
# ln -s /home/man/cat7 cat7
# ln -s /home/man/man1 man1
...
# ln -s /home/man/man7 man7
# ls -l /usr/share/man
total 82
drwxr-xr-x 2 bin  bin    512 Dec 19 13:08 ./
drwxr-xr-x 5 bin  bin    512 Feb 10 13:09 ../
lrwxrwxrwx 1 root other   12 Dec  6 09:41 X11 -> /opt/man/X11
lrwxrwxrwx 1 root other   14 Dec  6 10:26 cat1 -> /home/man/cat1
...
lrwxrwxrwx 1 root other   14 Dec 18 09:47 cat7 -> /home/man/cat7
lrwxrwxrwx 1 root other   14 Dec  6 10:26 man1 -> /home/man/man1
...
lrwxrwxrwx 1 root other   14 Dec 18 09:48 man7 -> /home/man/man7
#
```

The symbolic link to the */opt/man/X11* directory shown in this example was set up automatically by the OSF/Motif software's installation routine.

2.4.4 Named pipes

Since UNIX SVR2.2, communication between two processes has been supported by the named pipe or FIFO (first in - first out) mechanism. Conventional pipes can join a maximum of two processes to a parent process (e.g. *ls -l | more*). FIFOs by contrast are given a name by which they can be accessed throughout the system. For example:

```
# mknod /tmp/fifo p
# ls -l /tmp
total 20
drwxrwxrwt   3 ebt     sys         512 Oct 24 00:00 ebtpriv
prw-------   1 root    other         0 Oct 24 15:16 fifo
drwxrwxrwx   2 root    root       8192 May 28  1993 lost+found
-rw-rw-r--   1 root    sys         432 Oct 19  1993 sa.adrfl
# p=/tmp/fifo export p
# vi file1          # Switch to a second screen
vt1>vi file2        # Switch back to the first screen
:.,200 w >> $p      # Redirect the contents of the file from the
# current position through line 200 to the
# named pipe
```

```
# Switch to the second screen
:r $p               # Read in the contents of the named pipe
...
```

2.4.5 Special files

In Reliant UNIX, all accesses to devices (hard disks, floppy disks, terminals and so forth) are handled by device special files. Put simply, that means that even devices are treated as files. This makes it possible to apply Reliant UNIX commands to devices as well. The following command lists the contents of the files backed up on a tape in the tape drive:

```
# cat /dev/ios0/rstape003 | more
```

When a command or a program reads from a special file, what happens in effect is that the kernel fetches the data from the associated device. When data is written to a special file, the kernel sends the data to the device. The special file itself does not contain any data.
There is a distinction among special files based on the different methods of accessing the associated devices. For hard disks and partitions the distinction is as follows:

- Special files for buffered access (block mode):
  Accessing is handled in fixed-length blocks buffered in a disk cache. This access method is also known as block mode.
- Special files for unbuffered access (character mode):
  Accessing here is also block by block but unbuffered (raw mode). This access method is also referred to as character mode. This term is inaccurate, but since it is the term generally used in UNIX literature, it is used in this book as well.

Some special files exist in both character and block forms, because there are commands which support only one method of access. To mount a floppy disk as a file system using the *mount* command, for example, you have to use the block special file. Once the floppy disk is mounted, the file system cannot be checked with the *fsck* command, because *fsck* operates in character mode.
Reliant UNIX special files reside in the */dev* directory and its subdirectories.

```
# ls -l /dev | grep "^d"
drwxrwxrwx  2 root     root       512 Jul  7 09:51 X
drwxr-xr-x  2 root     root       512 Jul  7 09:49 abi
drwxr-xr-x  2 root     sys        512 Nov 22 13:43 at
drwxrwxrwx  2 root     root       512 Jan  4 10:12 dlpi
drwxr-xr-x  2 bin      bin        512 Nov 22 13:43 dptg
...
drwxr-xr-x  2 root     sys        512 Jul  7 09:49 rmt
drwxr-xr-x  2 root     sys        512 Nov 22 13:43 sad
drwxrwxr-x  2 root     root      1024 Nov 22 10:02 sxt
drwxr-xr-x  2 root     root       512 Nov 22 13:43 term
drwxr-xr-x  2 root     root      1536 Nov 22 13:43 vd
#
```

Special files are system-dependent. On a Reliant UNIX system you can run *autoconf -l* to list the installed devices. The *gettypes -R* command returns the SCSI devices supported for the current release; *gettypes -A* shows the SCSI devices in the current configuration.

```
# gettypes -A disk
MP65  disk  1278 MB  /dev/ios0/rsdisk000 disk0000
MP64  disk  1274 MB  /dev/ios0/rsdisk001 disk0001
MP64  disk  1274 MB  /dev/ios0/rsdisk002 disk0004
MP34  disk  585 MB  /dev/ios0/rsdisk022 disk0007
OS25 (CD-ROM)  cdrom  /dev/ios0/rsdisk006 disk0002
```

The *autoconf* command is described in the "System Administrator's Reference Manual" "[7]"; *gettypes* is documented in "Commands" "[5]".
The relationship between devices and special files is also described in the */etc/device.tab* file. This file is used by applications which need device-specific information, but it is also very useful for the system administrator. The entries relating to block and character special files are particularly helpful.
The structure of the file is as follows. The first element in each entry is an alias name (such as *diskette0* or *mdens0HIGH*). Then attributes are assigned to the specified device. In the example below you can see that the 3.5-inch floppy disk drive (*diskette0)* is accessed via the */dev/at/flp/rf0t* special file, the entire hard disk (*disk0*) via */dev/ios0/rsdisk000*, the partition containing the */usr* file system via */dev/ios0/rsdisk000s3*, and the cartridge tape drive *(smc0)* via */dev/ios0/rstape00* 3.

```
# pg /etc/device.tab
diskette0:/dev/at/flp/rf0t:/dev/at/flp/f0t::type="diskette" desc="3.5 inch Floppy
0" capacity="2880" controller="0" scsi="0" hwtype="- 3.5\" DS/HD Floppy Disk D
rive -    " display="true" removable="true" volume="diskette"
mdenslist="mdens0HIGH,mdens0LOW" mdensdefault="mdens0HIGH"
fmtcmd="/usr/sbin/flformat -v /dev/at/flp/rf0t"
erasecmd="/usr/sadm/sysadm/bin/floperase /dev/at/flp/rf0t" copy="true"
mkdtab="true"
...
disk0:/dev/ios0/rsdisk000:/dev/ios0/sdisk000:/dev/ios0/sdisk000:
dpartlist="s10,s0,s1,s2,s3,s4,s5,s15" type="disk" desc="Disk Drive 0"
capacity="2060910" controller="0" scsi="0" hwtype="- MP81 -SEAGATE ST31200N
858800004976  " display="true" part="true" mkdtab="true" removable="false"
displaycmd="/usr/sadm/sysadm/bin/dispdisk /dev/ios0/rsdisk000s7"
...
disk0s3:/dev/ios0/rsdisk000s3:/dev/ios0/sdisk000s3::mountpt="/usr" fstype="ufs"
dparttype="fs" type="dpart" desc="Disk Partition 3 of disk0" capacity="348840"
mkdtab="true" display="true" removable="false"
...
smc0:/dev/ios0/rstape003:::type="ctape" desc="Cartridge Tape 0" capacity="100000
0" controller="0" scsi="6" hwtype="- MC12 -TANDBERG TDC 3800 =04\:08CREATED02209
2 " display="true" removable="true" volume="Cartridge Tape" erasecmd="/usr/sadm/
mkdbin/tapecntl /dev/ios0/rstape003" mkdtab="true" norewind="/dev/ios0/rstape003
n" optionlist="smc0n,smc0h,smc0hn,smc0o,smc0ho"
```

The device entries are usually generated by the device installation procedure. If this is not the case, you must create the entries yourself. To add entries to the */etc/device.tab* file, and to remove or modify existing entries, you use the *putdev* command.
The *getdev* command produces a list of the devices in the */etc/device.tab* file and allows you to select devices according to defined criteria. In the following example *getdev* is used to list all devices with an *fstype* attribute value of *vxfs*.

```
# getdev fstype=vxfs
disk0000s2
disk0000s3
disk0000s4
disk0000s5
```

The *devattr* command lists the attribute values for a device:

```
# devattr -v diskette0
alias='diskette0'
bdevice='/dev/at/flp/f0t'
```

```
capacity='2880'
cdevice='/dev/at/flp/rf0t'
controller='0'
copy='true'
desc='3.5 inch Floppy 0'
display='true'
erasecmd='/usr/sadm/sysadm/bin/floperase /dev/at/flp/rf0t'
fmtcmd='/usr/sbin/flformat -v /dev/at/flp/rf0t'
hwtype='- 3.5" DS/HD Floppy Disk Drive - '
mdensdefault='mdens0HIGH'
mdenslist='mdens0HIGH,mdens0LOW'
mkdtab='true'
removable='true'
scsi='0'
type='diskette'
volume='diskette'
```

Note:

There are detailed descriptions of the *devattr*, *getdev* and *putdev* commands in the "System Administrator's Reference Manual" "[7]".

As the special file names are very cryptic, as system administrator you can create links to make it easier for yourself and your users to deal with special files. The drives can then be referenced by more sensible names.

```
# cd /dev/at/flp
# ln rfx3d floppy0
# ln rfx3h floppy1
#
```

On RM200/RM300/RM400 systems the IOS (Input/Output Subsystem) allows the following SCSI devices to be attached: hard disks, CD-ROM drives, tape drives and jukeboxes. RM600 systems support all these devices and floppy disk drives as well.

2.4.6 Hard disk special files

The names of the special files for SCSI hard disk drives and CD-ROM drives are of the following form:

```
/dev/ios0/[r]sdiskccusp
```

Code	Explanation
r	Specifies the character device. If *r* is not specified, the device is block-oriented
cc	Decimal (logical) controller number (channel ID): RM200, RM 300, RM400: 00 to 99 RM600: always 0
u	SCSI-ID (device unit number). Numbers 0 through 15 are possible, the values are given in hexadecimal format
z	Partition number (0-15, where 0-9 are specified as a one-digit decimal number and 0-15 must be

	specified as a two-digit decimal number).

Table 4: Hard disk and CD-ROM special files

2.4.7 Special files for floppy disk drives

The special files for floppy disk drives on RM200, RM300 and RM400 systems reside in the */dev/at/flp* directory. The following table shows the allocation of default special file names to the major floppy disk formats:

Special file	Floppy disk format
[r]f0	Automatic detection of formatted floppy disk type
[r]f0t	Automatic detection of formatted floppy disk type; references the entire disk (including cylinder 0)
[r]fx3d	3.5-inch, double-density (1 MB, 720K formatted)
[r]fx3dt	3.5-inch, double-density (1 MB, 720K formatted); references the entire disk (including cylinder 0)
[r]fx3h	3.5-inch, high-density (2 MB, 1.44 MB formatted).
[r]fx3ht	3.5-inch, high-density (2 MB, 1.44 MB formatted); references the entire disk (including cylinder 0)

Table 5: Floppy disk drive special files

The names of the special files for SCSI floppy disk drives on RM600 systems are of the following form:

```
/dev/ios0/[r]sfdisk00yf[3d|3h|3e][t]
```

Code	Explanation
r	Specifies the character device. If *r* is not specified, the device is block-oriented
y	Number of the SCSI floppy disk drive on a SCSI bus (0-6)
f[d|h|e]	Floppy disk type; the following types are available: d - 3.5-inch, double-density (1 MB, 720K formatted) h - 3.5-inch, high-density (2 MB, 1.44 MB formatted) e - 3.5-inch, extra-high-density (4 MB, 2.88 MB formatted)

	If the type is left unspecified, it is detected automatically.
t	If *t* is specified, the entire floppy disk (including cylinder 0) is referenced.

Table 6: SCSI floppy disk drive special files

2.4.8 Special files for cartridge tape drives (streamers)

The names of the special files for SCSI cartridge tape drives are of the following form:
`/dev/ios0/[r]stapeppu[c][h][n][o][v][g]`

Code	Explanation
r	Specifies the character device. If *r* is not specified, a block-oriented device is used
pp	Logical controller number
u	Number of the SCSI drive (device unit number). Numbers 0 through 15 are possible, the values 10 to 15 are represented in hexadecimal format
n	If specified, the tape is not automatically rewound
h	If specified, a high recording density is used
c	If specified, the data is compressed
o	Selects the OSX device which is not to be rewound when closed
v	If specified, variable block mode is used
g	If specified, the generic *ioctl* port is used for reading and writing via *ioctl(2)* system calls in the tape driver

Table 7: SCSI cartridge tape drive special files

3 Reliant UNIX startup and shutdown

This chapter describes what happens when the operating system is started up and shut down. As system administrator, an understanding of these procedures will put you in a better position to analyze possible malfunctions and to allow users to work more efficiently by making all the important functions available automatically.

3.1 System run levels

A Reliant UNIX system can be in more states than simply "On" and "Off". When a system is switched on, it can be in a number of different states (off, of course, stays off).
In UNIX terminology, these states are known as run levels. While the system is up and running, you can use the *who -r* command to find out what run level it is on. When it is switched off, it is on run level 0. In the example below, the system is on run level 2. That means that the normal multiuser mode is active and, in addition, that exported file systems on remote hosts can be accessed and/or remote hosts are allowed to access local imported file systems; table "System run levels" lists the available run levels. The run level numbers are used in the options of the *init* and *telinit* command and in the *shutdown* script (see section "Switching run levels").

```
# who -rH
NAME       LINE         TIME          IDLE    PID  COMMENTS
.       run-level 2  Jul 18 09:32    2       0    S
#
```

As a rule, once the system has been booted it is on either run level 2 or 3 (the initial run level is determined by the *initdefault* parameter in the */etc/inittab* file, see section "The /etc/inittab file"). For error recovery and data backup it may be necessary to switch to single-user mode (see section "Switching run levels").

Run level	Explanation
run level 0	The operating system is being shut down, which means that all processes are being terminated and the system can be switched off or will switch off automatically.
run level 1	Single-user mode. All processes other than those which are running unattached to any terminal (daemons) and those which are associated with the console are terminated. File systems remain mounted, but users cannot log in.
run level 2	"Normal" multiuser mode. On Reliant UNIX systems the functions of run level 3 are integrated in run level 2.
run level 3	Multiuser mode plus exported and imported file systems.

run level 4	"Configured" multiuser mode. Can be used by the system administrator to set up a predefined range of functions, with specific programs being made available or unavailable. Rarely used.
run level 5	Shutting down to ROM mode, or terminating all processes and rebooting the system.
run level 6	Terminating all processes and rebooting the system.
run-level a,b,c	Configurable add-ons for launching extra applications.
run level s or S	Similar to run level 1, but with a password check, as it is possible for a terminal other than the console to be used. There are no file systems mounted and no software is loaded. (Does not work from a terminal emulator or an X terminal, because these are both pseudo terminals.)
run level q or Q	Rereading the */etc/inittab* file after modifications. Modifications do not take effect until this has been done.

Table 8: System run levels

3.2 System startup procedure

When you switch on the computer, a series of routines is carried out to make the computer and the operating system operational. This procedure is known as booting. It is to some extent processor-dependent.

3.2.1 Booting RM200, RM300 or RM400

When you switch on the system, the firmware runs automatically. It checks the hardware and logs the results on the console. This base-level system initialization is followed by the loading of the Prom monitor (or the firmware monitor on a computer with graphics capability).
The Prom monitor first runs the startup script (processor availability check, RAM availability check, peripheral interface test, SCSI device availability check) and logs the results of the tests on the console (the exact procedure being governed by the *bootmode* environment variable). The Prom monitor features a set of system configuration commands which are available even if the operating system cannot be accessed. For details of the Prom monitor's command set refer to the *prom(8)* entry in the "System Administrator's Reference Manual" "[7]".
The following is an example of the messages generated by the power-on tests.

```
Running Power-On Diagnostics...
Processor 01
Cache Test #1...PASSED
Secondary Cache Test...PASSED
ECC Cache Test...PASSED
Data Cache MATS+ Test...PASSED
FP Test #1...PASSED
FP Test #2...PASSED
Processor 0
Cache Test #1...PASSED
Secondary Cache Test...PASSED
ECC Cache Test...PASSED
Data Cache MATS+ Test...PASSED
FP Test #1...PASSED
FP Test #2...PASSED
Memory Test...PASSED
yMemory Test...PASSED
SCSI CTLR NCRC810 ONBOARD DEVICE 01:
Read/Write Test...PASSED
Inc/Dec Test...PASSED
SCSI FIFO Test...PASSED
DMA FIFO Test...PASSED
SCRIPT Interrupt Test...PASSED
DMA copy Test ...PASSED
...
IdProm Test...PASSED
Ethernet address: 00:00:e4:03:00:0e:
Ethernet Checksum: 0x00
LAN AMD79C970 CONTROLLER: 00
Ethernet Address...PASSED
Port test...PASSED
Lan Internal LoopBack Test...PASSED
```

```
Lan External LoopBack Test...PASSED
EISA ICU Test...PASSED
Duart 452/552 Port Tests...Channel B...PASSED
Centronics Port Test...PASSED
8254 Timer 00 Test...PASSED
8254 Timer 02 Test...PASSED
Time-of-Day Clock Test...PASSED
Ending Power-On Diagnostics...
Machine is coming up...
```

The Prom monitor then loads the stand-alone shell (SASH) into RAM. The SASH is a shell which is independent of the operating system and like the Prom monitor features a series of user commands which can be executed regardless of the presence of the operating system. It is on partition 10 of the system disk and has comparable functionality to Boot 2 on the RM600. The SASH loads the operating system from a defined medium and on completion of the boot procedure passes control to the operating system. The functionality of the SASH is documented in the *sash(8)* entry in the "System Administrator's Reference Manual" "[7]". The following is an example of the output produced while Reliant UNIX is being loaded.

```
Loading dkncr(0,0,0)unix
3868004+2189328+1406376 entry: 0x80030100
SNI  RM400
UNIX(R) SINIX-N Release 5.44 Version A0020
Copyright (c) Siemens Nixdorf Informationssysteme AG 1990-1997
Basis: DC/OSx (R), Copyright (c) Siemens Pyramid Information
Systems, Inc. 1984;
UNIX (R), Copyright (c) X/Open Company Limited 1983
All rights reserved
SINIX is a registered trademark of Siemens Nixdorf Informationssysteme AG
DC/OSx is a registered trademark of Siemens Pyramid Information
Systems, Inc.
Reliant is a registered trademark of Siemens Pyramid Information Systems, Inc.
X/Open is a registered trademark, and the X device is a trademark,
of X/Open Company Limited.
UNIX is a registered trademark in the United States and other countries,
licensed exclusively through X/Open Company Limited.
Processor Configuration:
Boot  processor:   Cpu #0 (p0)
Other processor(s):    #1 (p1)
...
```

Note:

For further information on the boot procedure for an RM200/RM300/RM400 refer to the manual "Installing the Operating System (RM200/RM300/RM400)" "[2]".

3.2.2 Booting an RM 600-xxx, RM600-E

After it has been powered up, the system performs a number of self tests. While this is going on, dots are output on the console. Then the Master Test Handler (MTH) detects the system configuration and logs it on the console.
Having completed its work, the MTH automatically starts up the Board Debug Monitor (BDM). The BDM is a monitoring program and, in the case of an RM600xxx system, resides

on the CSI controller (CSI stands for Central Services & Interfaces) or, in the case of an RM600-E system, on the HIOS controller (High Performance Output System). It features a series of user commands which are available regardless of the presence of the operating system.
The BDM next loads and runs Boot 1 (first-level boot), which in turn runs Boot 2 (second-level boot). Boot 2 resides on partition 10 of the system disk and has comparable functionality to the SASH on an RM400. Boot 2 in turn loads an executable Reliant UNIX kernel into RAM, starts it up and passes control to it. Boot 2 is then deactivated. Like the SASH, Boot 2 features a series of interactive user commands. The Boot 2 command set is summarized in the *boot2(8)* entry in the "System Administrator's Reference Manual" "[7]".

Note:

For further information on the boot procedure for an RM600-xxx refer to the manual "Installing the Operating System (RM600-xxx, RM600-E)" "[3]".

3.2.3 The /etc/inittab file

The first process started on completion of internal initialization is the scheduler. This is responsible for allocating CPU time to individual processes under the Reliant UNIX multitasking mechanism (see table "Important processes and their functions"). Then the */sbin/init* process is started. One of its main duties is to schedule other processes which guarantee the functioning of the operating system (see section "Processes"). The *init* process continues running for as long as the operating system does. It fetches its instructions from the */etc/inittab* file.

```
# cat /etc/inittab
ap::sysinit:/sbin/autopush -f /etc/ap/cons.ap
ak::sysinit:/sbin/wsinit >/dev/null 2>&1
bchk::sysinit:/sbin/bcheckrc </dev/console >/dev/console 2>&1
sofs::sysinit:/sbin/soconf < /dev/console >/dev/console 2>&1
brc:1234:bootwait:/sbin/brc >/dev/console 2>&1 </dev/console
is:2:initdefault:
rS:S:wait:/sbin/rcS >/dev/console 2>&1 </dev/console
r0:0:wait:/sbin/rc0 off > /dev/console 2>&1 </dev/console
r1:1:wait:/sbin/rc1 >  /dev/console 2>&1 </dev/console
r2:23:wait:/sbin/rc2 > /dev/console 2>&1 </dev/console
r3:3:wait:/sbin/rc3 >  /dev/console 2>&1 </dev/console
r5:5:wait:/sbin/rc0 reboot >/dev/console 2>&1 </dev/console
r6:6:wait:/sbin/rc6 reboot >/dev/console 2>&1 </dev/console
sd:0:wait:/sbin/uadmin 2 0 >/dev/console 2>&1 </dev/console
fw:5:wait:/sbin/uadmin 2 2 >/dev/console 2>&1 </dev/console
rb:6:wait:/sbin/uadmin 2 1 >/dev/console 2>&1 </dev/console
li:23:wait:/usr/bin/ln /dev/systty /dev/syscon >/dev/null 2>&1
sc:234:respawn:/usr/lib/saf/sac -t 300
co:1234:respawn:/usr/lib/saf/ttymon -g -p "Console Login: " -d /dev/ttylc -l
console
pfsh:s0123456:powerfail:/sbin/powerfail.sh >/dev/console 2>&1
#
```

There are four parts (fields) to an entry in the */etc/inittab* file, separated by colons. The *init* process evaluates the entries line by line. The information which appears in the various

fields is as follows:

Field 1:
The first field in each entry is a unique label, such as *r3*. The label is for identification purposes only and has no other function.

Field 2:
The second field defines the run level the entry applies to. There may also be a list of run levels, entered one after the other without separators. If the field is empty, the entry applies to all run levels.

Note:

The second field in the entry labeled *is* must never be empty. If it is, an endless loop will be created, with the system being continually restarted without ever becoming operational.

Field 3:
The third field defines the action the *init* process is to take when the system is on the run level indicated in field 2.

Action	**Meaning**
boot	Execute only once on operating system startup and the first time the */etc/inittab* file is read.
bootwait	Execute only once on operating system startup and the first time the */etc/inittab* file is read, and wait for the command in field 4 to complete before examining the next entry.
initdefault	Set the initial run level (see section "System run levels").
off	Terminate the process specified in the next field if it is running.
once	Run the program specified in the next field without waiting for the process to complete.
powerfail	Execute the specified program in the event of power supply problems or overheating.
respawn	Restart the specified program whenever */sbin/init* detects that it has completed.
sysinit	Execute only once on operating system startup before the *Console Login:* message appears on the screen.
wait	Execute the program specified in the next field and wait for the process to complete before examining the next entry.

Table 9: Actions in field 3 of the /etc/inittab file

Field 4:

The fourth field contains the name of the command or shell script to run for the associated run level. The following table explains the entries in the sample */etc/inittab* file illustrated above.

Command/script	**Meaning**
/sbin/autopush	Configure STREAMS driver
/sbin/wsinit	Create console devices and initialize graphical console
/sbin/bcheckrc	Check *root* and *var* file systems and mount pseudo file systems */proc* and */dev/fd*
/sbin/soconf	Configure kernel socket database using */etc/netconfig*
/sbin/brc	Initialize I/O system and virtual disks
/sbin/rcS	Run script to change to single-user mode (run level S)
/sbin/rc0	Run system shutdown script
/sbin/rc1	Run script to change to single-user mode (run level 1)
/sbin/rc2	Run script to change to multiuser mode
/sbin/rc3	Run script to change to multiuser mode with extended networking functionality
/sbin/rc6	Run script to reboot the operating system
/sbin/uadmin	Internal system call used to switch run levels
/usr/lib/saf/sac	Start the Service Access Controller (see also section "The rc files")
/usr/lib/saf/ttymon	Process causing the *Console Login:* message to be displayed. Replaced by a command shell following successful login and restarted when the shell is terminated.
/sbin/powerfai	Run script to initiate shutdown in

l.sh	response to power supply problems or overheating

Table 10: Entries in field 4of the /etc/inittab file

At boot time the */etc/inittab* file described in section "The /etc/inittab file" is processed as follows: first lines 1 through 6 are read and executed, then line 10 and finally lines 17 through 19.
As a rule the system administrator has no need to make any changes to the file, as the file installed after operating system installation is fully functional. One exception to this rule might be the line defining the default run level (*initdefault*) if run level 3 is required instead of run level 2.

3.2.4 The rc files

Depending on the run level defined in the */etc/inittab* file, the correspondingly named */sbin/rc?* script (e.g. */sbin/rc2*) executes the matching rc (run command) files from the correspondingly named */etc/rc?.d* directory (e.g. */etc/rc2.d*).
The files in */etc/rc?.d* are scripts designed to start and kill programs and processes, respectively named *S[0-9][0-9]** (for example */etc/rc2.d/S03savecore*) and *K[0-9][0-9]** (for example */etc/rc2.d/K20nfs*). As they are all ASCII files, it is easy for the system administrator to gain a good idea of exactly what happens.
The following example illustrates the order in which the scripts are run when the system boots to multiuser mode, which is run level 2.

```
# cd /etc/rc2.d
# ls -l S*
-rw-r--r-- 3 root   sys     1794  Jun 27  1996 S01MOUNT0FSYS
-rw-r--r-- 2 bin    bin      392  Jun 27  1996 S02PRESERVE
-rwxr-xr-x 3 root   other    909  Jun 27  1996 S02errord
-rw-r--r-- 2 root   sys     1682  Jun 27  1996 S02savedump
-rwxr-xr-x 3 root   other   2505  Jun 27  1996 S03swap
-rw-r--r-- 3 root   sys      223  Jun 27  1996 S04MOUNT1FSYS
-rw-r--r-- 2 root   sys      809  Jun 27  1996 S05RMTMPFILES
-rw-r--r-- 3 root   sys      764  Jun 27  1996 S09osm
-rwxr--r-- 3 root   sys      554  Jul 13  1996 S12et
-rw-r--r-- 2 root   sys      496  Jun 27  1996 S13op_disk
-rw-r--r-- 2 root   sys     1570  Jun 27  1996 S15mkdtab
-rw-r--r-- 1 root   other    161  Jun 27  1996 S18setuname
-rwx------ 2 root   sys      394  Jun 27  1996 S18silsd
-rwxr-xr-x 1 root   sys      628  Jun 27  1996 S19tt_open
-rwxr--r-- 2 root   sys     1308  Jun 27  1996 S20sysetup
-rwxr--r-- 2 bin    bin      680  Jun 27  1996 S21perf
-rwxr--r-- 1 root   other   1946  Jun 24  1996 S26log3
-rw-r--r-- 2 root   sys     6573  Jul 13  1996 S51uts
-rwxr--r-- 4 root   other   1653  Feb 21  1996 S52IOCS30
-rw-r--r-- 4 root   sys      659  Jun 27  1996 S65syslog
lrwxrwxrwx 1 root   root      20  Nov 22 10:57 S69inet -> /etc/init.d/inetinit
lrwxrwxrwx 1 root   root      17  Nov 22 11:21 S70SIlvs -> /etc/init.d/SIlvs
-rwxr-xr-x 4 root   sys       33  Aug 16  1996 S70termadm
-rw-r--r-- 2 root   sys      456  Jun 27  1996 S70uucp
-rwx------ 2 root   sys     1305  Jun 27  1996 S71SAFEchk
-rw-r--r-- 5 root   sys      763  Jun 27  1996 S75cron
-rwxr-xr-x 5 root   sys     3115  Jun 27 12:00 S75rpc
```

```
-rwxr-xr-x 4 root   sys     4820  Jun 27  1996 S76yp
-rwxr--r-- 3 root   bin      833  Jun 16  1996 S77snmpd
-rwxr-xr-x 5 root   sys     2978  Jun 27  1996 S81nfs
-rw-r--r-- 5 bin    bin     1948  Jun 27  1996 S88smtpd
-r-xr-x--- 3 bin    sys     4685  Mar 23  1996 S91SIspl
-rw-r--r-- 2 root   sys      883  Jun 27  1996 S96sandm
lrwxrwxrwx 1 root   root      20  Mai 22 11:16 S98conf_ana ->
/etc/init.d/conf_ana
-rwxr-xr-x 1 root   sys      777  Mar 16  1996 S99mocd
-rwxr-xr-x 4 root   sys     1561  Jun 13  1996 S99motif
-rw-r--r-- 2 root   sys      616  Jun 27 14:30 S99saf
#
```

The *S01MOUNT0FSYS* script, for example, checks and mounts all file systems, while *S18setuname* sets the host name or system name. The *S75cron* script starts up the cron service (see section "The cron service").
If necessary, as system administrator you can also stop and restart these scripts individually. This may be called for if, for example, the AT&T spooler cannot be terminated in the normal way or if there are problems with the mail system.

3.2.5 Port monitoring by the System Access Facility (SAF)

For Reliant UNIX 5.4x there have been considerable changes to the system access control mechanism. Previously, all login accesses, both from local terminals and over a network, were managed and controlled by the *init* process, which spawned two other processes to handle them, *getty* for local services and *listen* for network services. Now the System Access Facility (SAF) handles all those duties and replaces these two service types with services of its own.
Under the System Access Facility, the management and control of all connection services is assigned by the *init* process to individual services. The SAF consists of the Service Access Controller (SAC), which is a port monitoring control process, and the port monitors, which are responsible for the various services.
The Service Access Controller is started from the */etc/inittab* file by the *init* process when the system boots to multiuser mode (see section "The /etc/inittab file"). The SAC in turn starts up the port monitors, which monitor specific ports for accesses to the system.
The port monitoring model can be schematized in the following way:

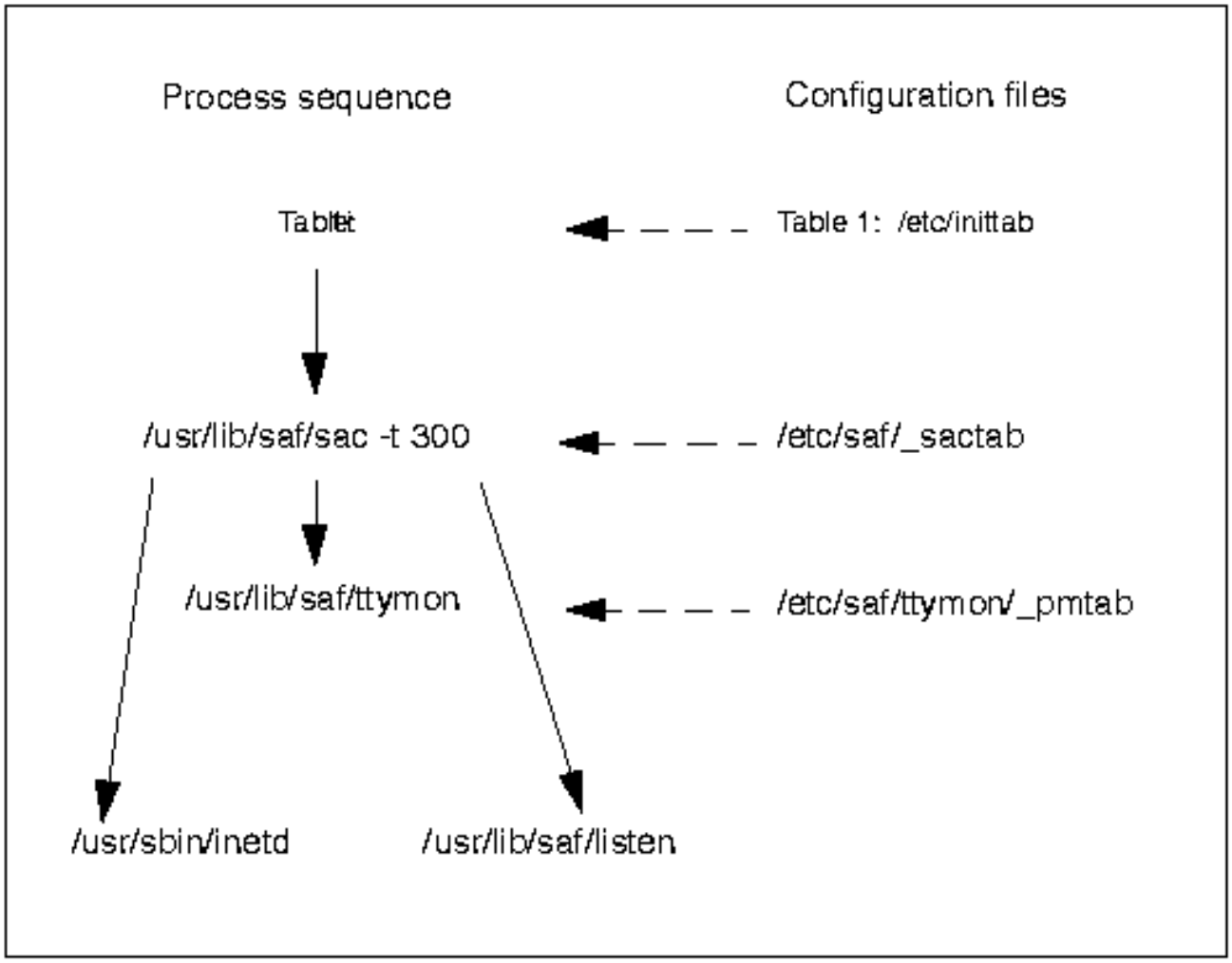

Figure 5: Port monitoring model

The */etc/saf/_sactab* configuration file is configured using the system administration interface or the *sacadm* command.
The */etc/saf/ttymon/_pmtab* configuration file is configured using the system administration interface or the *pmadm* or *ttyadm* command (see section "The sysadm interface" and section "Configuring terminals").

Port monitor	**Function**
ttymon	Monitors and controls access to local terminals
inetd	Monitors and controls network access to remote services (commands such as *rlogin* , *rcp* and *rsh*)
listen	Monitors and controls network access to other network services (remote procedure calls, etc.)

Table 11: Reliant UNIX port monitors

The configuration of the SAF, the SAC and the port monitors as it stands after operating system installation should initially be perfectly suitable. You can make changes later using the *sysadm* interface or the appropriate shell commands, paying close attention to the accompanying documentation.

3.2.6 The /etc/vfstab file

The */etc/vfstab* file contains a list of the file systems that are to be mounted in the directory hierarchy when the system is booted. It includes both local file systems and file systems on remote hosts. At system startup, all the file systems listed in */etc/vfstab* are checked by *fsck* and then mounted. The file has the following format:

```
# cat /etc/vfstab
# block                fsck                  mount     fs  fsck auto mount
# dev                  dev                   point    type pass mnt  opts
/proc                  -                      /proc   proc  -   -    rw
/dev/fd                -                      /dev/fd fdfs  -   -    rw
/dev/ios0/sdisk000s0 /dev/ios0/rsdisk000s0   /       ufs   -   -    rw
/dev/ios0/sdisk000s1 /dev/ios0/rsdisk000s1   -       swap  -   -    rw
/dev/ios0/sdisk000s2 /dev/ios0/rsdisk000s2   /opt    vxfs  0   yes  rw
/dev/ios0/sdisk000s3 /dev/ios0/rsdisk000s3   /usr    vxfs  0   yes  rw
/dev/ios0/sdisk000s4 /dev/ios0/rsdisk000s4   /var    vxfs  0   yes  rw
/dev/ios0/sdisk000s5 /dev/ios0/rsdisk000s5   /home   vxfs  0   yes  rw
#
```

Field	Meaning
Field 1	Block special file needed for the *mount* command
Field 2	Character special file needed for the *fsck* command
Field 3	File system mount point
Field 4	File system type
Field 5	Pass number for the *fsck* command
Field 6	File system automount option
Field 7	Options for the *mount* command

Table 12: Contents of the /etc/vfstab file

The */etc* directory also contains the *mnttab* file. This file is a record of the file systems which are currently mounted. Editing this file has no effect on system behavior but only falsifies the output of the *mount* command. Conversely, changes to the */etc/vfstab* file take effect the next time the system is booted or the *mountall* command is issued.

3.3 Switching run levels

It may be necessary to switch to another run level if, for example, you need to make a backup or perform administrative tasks or shut down the system for the weekend. There are a number of ways to switch to another run level. You can use the *init* , *uadmin* or *telinit* command, the *shutdown* shell script or the *sysadm* interface.

Warning:

> You should not turn off the power switch or pull out the power plug unless there is an emergency, for example if life is at risk. There are always processes running on a Reliant UNIX system, and these must be properly terminated first.

You can look at the */etc/inittab* file to find out which commands are executed when the required run level is switched to (see section "The /etc/inittab file").
The following examples illustrate a number of ways of switching run levels:

- The best way to switch to run level 0, system shutdown and power-off, is to run the */sbin/shutdown* script. First you must change directories to */* (*root*), as otherwise the script will abort and issue an error message.
  For security reasons you should call the */sbin/shutdown* script with a delay (grace period) of at least 30 seconds (in other words, the value of the *-g* option should not be under 30; if you call */sbin/shutdown* without any options, the grace period is 60 seconds). For details of what happens when you switch to run level 0, refer to section "Reliant UNIX shutdown procedure".

```
# cd /
# shutdown -y -g30 -i0
# The following message is sent to
# all users who are logged in:
# The system will go down in 30 seconds
# Please log off now
Shutdown started.    Tue Jul 08 16:36:46 MET 1997
...
...
Changing to init state 0 - please wait
...
#
```

- You can switch to single-user mode from any terminal by using the *init* command.

```
# init s
#
```

- To shut down the operating system quickly without warning your users beforehand you could enter the following series of commands. (Note that this method should be used only in exceptional circumstances, as users who are logged in are given no chance to exit their programs correctly.) The *sync* command causes all data in the buffer to be written out to hard disk before the system goes into run level 0. (Unlike the *init* command, the */sbin/shutdown* script invokes *sync* automatically.)

```
# sync
# umountall
# init 0
#
```

- The *uadmin* command uses the internal system call for graceful system shutdown. It is also invoked from the */etc/inittab* file when the computer is shut down (see section "The

/etc/inittab file").

```
# uadmin 1 0     #equivalent to init 0
# uadmin 2 0     #equivalent to sync;sync;init 0
#
# uadmin 1 1     #equivalent to init 6
# uadmin 2 3     #equivalent to sync;sync;init 0
#
```

- The *telinit* command is a link to the *init* command. It is described on the *init* man page (see the "System Administrator's Reference Manual" "[7]"). The following example causes the *init* process to reread the */etc/inittab* file after changes have been made to it.

```
# telinit q
#
```

When using the *sysadm* interface you can initiate a reboot or a shutdown with the menu items *reboot* and *shutdown* respectively. Both are on the *machine* menu.

sysadm interface		
Menu item	**Function**	**Shell command**
machine - reboot	Reboots the system	/sbin/shutdown
machine - shutdown	Shuts down the system	/sbin/shutdown

Table 13: Switching run levels with sysadm

3.4 Reliant UNIX shutdown procedure

There are essentially three actions that have to be taken when the system is shut down:

- Any data still in the buffer must be written out to the hard disk to avoid data loss
- All the file systems mounted on the *root* file system must be unmounted
- A flag must be set to suppress file system checking the next time the system is booted

These three actions are not always performed automatically. As already described in section "Switching run levels", there are a number of ways to shut down the operating system. The most reliable approach to performing the above actions is the */sbin/shutdown* script. Any users that are logged in are warned, all active processes are terminated, the contents of the buffer are written out to hard disk, the scripts in */etc/rc0.d* are executed, all file systems are unmounted, and the computer is then switched off.

```
# cd /
# shutdown -y -g60 -i0
```

4 System administration tools

The following chapter introduces the tools you need to perform your duties as system administrator. These include an effective working environment, system administration interfaces and a helpful shell, together with commands and scripts, though only a small selection of these can be covered here. Some commands not dealt with in this chapter are discussed in other chapters of the book. This applies especially to the login administration commands, described in section "Managing accounts and user groups", and the data backup commands, covered in section "Backing up and restoring data".

4.1 Working environment

You should first configure a working environment appropriate to your own needs. If you are working with an alphanumeric terminal, you can set up a custom environment in a *.profile* file for the *root* account.
The system administrator can also customize the system-wide working environment, as described in section "The /etc/skel directory" and section "The /etc/profile file".

```
# cat /.profile
PATH=$PATH:/usr/ucb:/bin
MANPATH=$MANPATH:/usr/share/man:/usr/share/man/X11
EDITOR=vi
FCEDIT=vi
HISTSIZE=10000
export PATH MANPATH EDITOR FCEDIT HISTSIZE
PS1="(`/usr/ucb/whoami`\@`/sbin/uname -n`)"'$PWD: '
LANG=En
export PS1 LANG
alias l='ls -laF'
alias ll='ls -laF|more'
#
```

The above example primarily illustrates how to set paths and define and export variables. At the end of the file there are some alias definitions, which are effective only in the Korn shell (see section "The Korn shell").
A further feature of the Korn shell is seen in the definition of the *PS1* shell variable, which affects the appearance of the primary prompt. On a computer with the host name *hamlet* the above definition would result in the following prompt string if you were in the */var/adm* directory and were logged in as *root*:

```
(root@hamlet)/var/adm:
```

Owing to a special function of the Korn shell, the current absolute path name appears in the prompt if *$PWD* is included in the *PS1* shell variable.
On computers with graphics capability, you can also use the functionality of the SINIX/windows User Environment. SINIX/windows is a working environment which allows you to access graphical applications on your monitor using a mouse or the keyboard (see also section "SINIX/windows"). SINIX/windows User Environment gives you the option of using the "SINIX Desktop" or the "Common Desktop Environment" (CDE). CDE complies with the X/Open specifications for the Common Desktop Environment and is the default working environment. (For this reason only CDE is described in the following sections).

SINIX/windows provides a range of applications and services, some of which can make system administration easier for you. The files required for configuring the working environment under SINIX/windows are described in the SINIX/windows documentation.

Note:

For further information on SINIX/windows refer to the manuals "SINIX/windows User Environment V3.0, User's Guide (CDE)" "[19]", and "SINIX/windows User Environment V3.0, Advanced User and System Administrator Guide (CDE)" "[21]", or the manuals "SINIX/windows User Environment V3.0, Introduction to Handling and Configuration (SINIX Desktop)" "[20]" and "SINIX/windows User Environment V3.0, Guide for Experts and System Administrators (SINIX Desktop)" "[23]".

4.2 The sysadm interface

The standard alphanumeric interface for system administrators under Reliant UNIX 5.4x is sysadm. This interface allows you to perform all the most important system administration tasks.
The range of functions offered by the interface may vary from system to system. Some application programs, for example, generate their own menu items, and the precise configuration of the hardware affects the hardware-specific items.
The examples below reflect only a small part of the extensive functionality of the sysadm interface.

Note:

For an in-depth description of the sysadm interface refer to the manual "System Administration and Hardware Configuration, SYSADM" "[10]".

The *sysadm* menu, message, command and help texts have been internationalized. The *LANG* environment variable can be used to select the language you require.
You can call the *sysadm* interface from the shell by entering the *sysadm* command or from the SINIX/windows desktop by selecting *MainDesktop* - *System* - *Administration* - *SysAdm*. This then opens the sysadm main window (which may differ in some details from what is shown below):

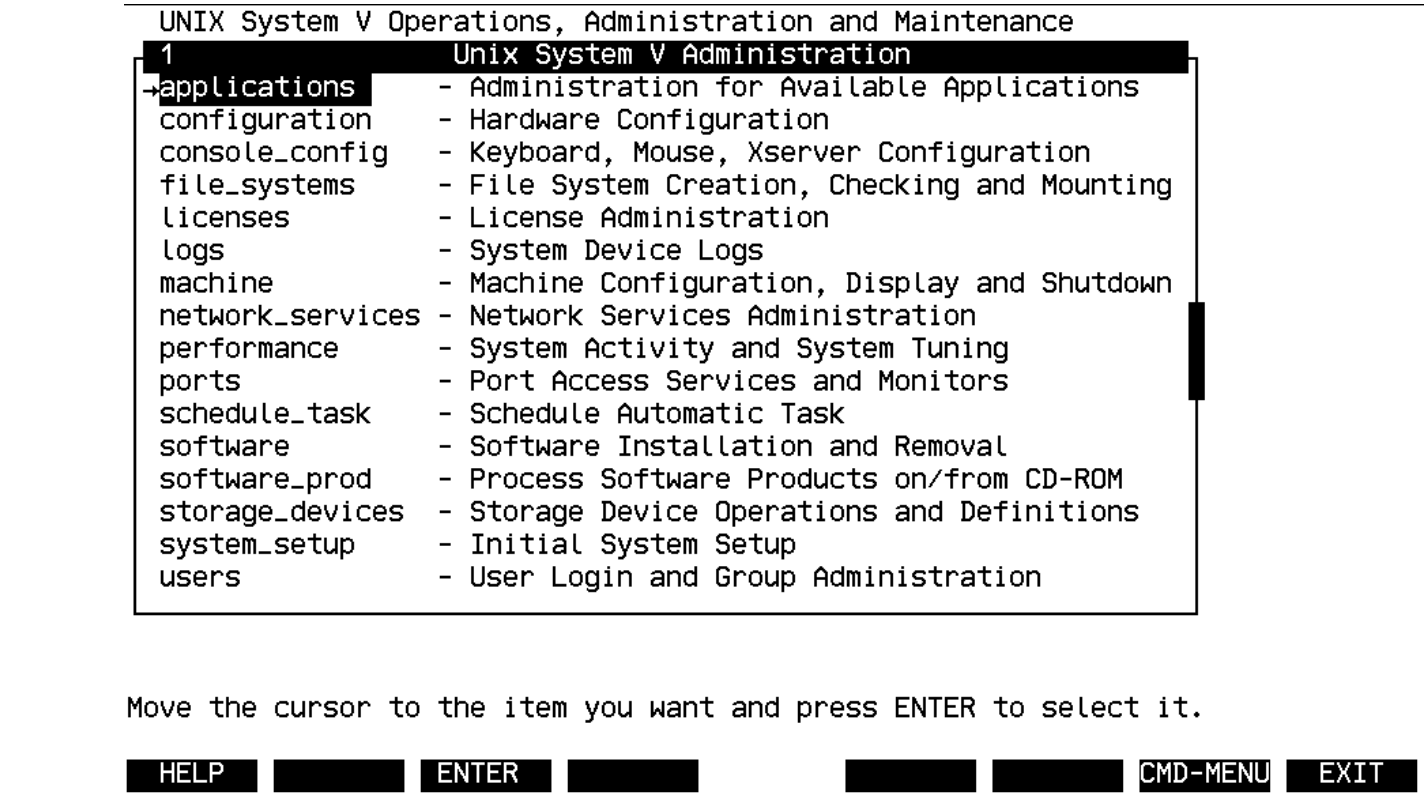

Figure 6: Typical sysadm main window

The main window is divided into five areas: the header line, the work area, the message line, the command line and the function key line.

Area	Function
Header line	The header line along the top of the window contains the program name. The message "working" appears in the top right-hand corner when *sysadm* is performing a task. No input is accepted while this message is being displayed.
Work area	The work area, which occupies the majority of the screen, displays menu windows or forms. This is where you select menu items, which mostly open further menu windows.
Message line	The message line immediately below the work area displays information and error messages.
Command line	The command line is only active if direct command input is supported in addition to menu item selection. In such cases a "-->" prompt is displayed
Function keys	Along the bottom of the window there are eight rectangles representing function keys **[F1]** through **[F8]**, which you can use to perform the indicated functions. The labeling and hence the functions of the keys may vary from menu to menu.

Table 14: Areas of the sysadm inteface window

The sysadm interface is keyboard-operated. To select menu items you can use the arrow keys **[UP]** **[DOWN]** or the key for the initial letter of the required menu item. To move to the next menu you press the **[F3]** key, which in the sysadm interface has the function of the **[RETURN]** key (you need only press **[RETURN]** if no mapping for *ENTER* or *CONT* is shown in the function key area).
The function key area indicates the current mapping of the function keys. The following table lists the 11 possible function key assignments. How the keys are mapped depends on the tasks that need to be performed.

Function key	Function
CANCEL	Quit current window and return to previous window
CHOICES	List available choices
ENTER	Select current menu item; close form
HELP	Display help information; the help that is shown depends on the current cursor position

CMD-MENU	Display command menus
MARK	Mark values on a CHOICES menu for selection
RESET	Reset the value at the current cursor position to its default
NEXTPAGE	Go to next page of text
PREVPAGE	Go to previous page of text
SAVE	Save entered values and if necessary open another form or initiate an action
CONT	Continue with a task interrupted for an acknowledgment

Table 15: Function key mapping in the sysadm inteface

The **[HELP]** function key calls up a help text for the selected menu item.
If there are no function keys on your keyboard, you can use the key combination **[CTRL]** - **[F]** *n* instead, where *n* is the number of the corresponding function key, e.g. **[CTRL]** - **[F] [3]** for **[F3].**
Forms in the sysadm interface comprise one or more data entry fields. Some forms have default values preset. You can often call up a list of the available values by pressing the *CHOICES* function key.
The following example illustrates the typical system administration duty of adding a new user:

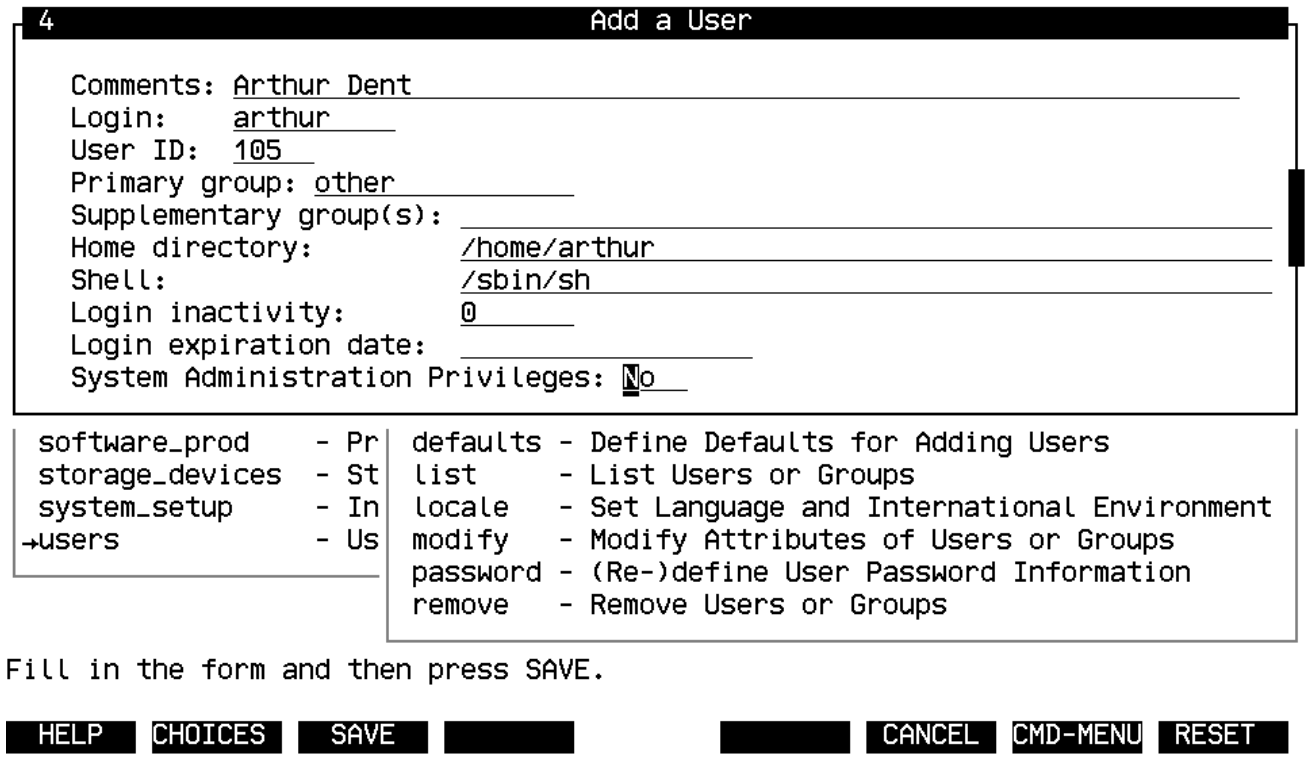

Figure 7: Adding a user with sysadm

The next two figures demonstrate system activity reporting (which can also be done with the *sar* command). First the *show_overview* menu item is selected, then the activities are displayed.

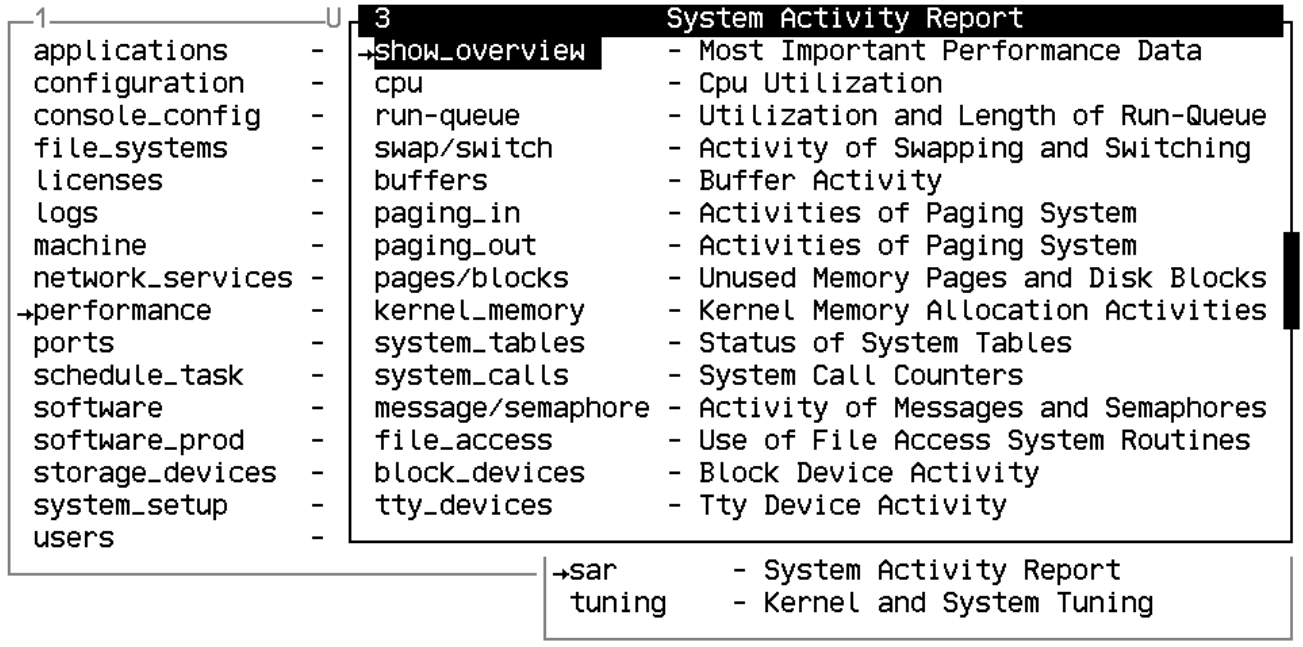

Figure 8: Selecting system activity reporting with sysadm (example)

```
ReliantUNIX-N sodom 5.44 A0020 RM400                08/01/97    15:04:01

---CPU---       --SWAPPING---    --SWITCHING--   ----MEMORY----  -FILE ACCESS-
%usr    0       swpin/s   0.0    pswch/s     5   freem     5064  iget/s       0
%sys    1       pswin/s   0.0    runq-sz  1.00   frees   231168  namei/s      0
%wio    0       swpot/s   0.0    %runocc    20   shmem        0  dirblk/s     0
%idle  99       pswot/s   0.0

--PAGING IN-    --PAGING OUT-    ----BUFFER---   --------SYSTEM CALLS----------
pgin/s   0.0    pgout/s   0.0    bread/s     0   scall/s     15  sread/s      1
ppin/s   0.0    ppgout/s  0.0    lread/s     0   fork/s    0.00  rchar/s   2638
pflt/s   0.0    pgfree/s  0.2    pread/s     0   exec/s    0.00  swrit/s      1
vflt/s   0.0    pgscan/s  0.0    bwrit/s     0   msg/s      0.0  wchar/s   2656
slck/s   0.0    %ufipf/s  0.0    lwrit/s     0   sema/s     0.0
                                  pwrit/s     0

---SYSTEM TBL---  --TTY DEVICE--  --DEVICE-%BUSY-R+W/S--
proc-sz       61  rawch/s      0  sdisk000           |
ov             0  canch/s      0  sdisk001
inod-sz     1063  outch/s     17  sdisk005
ov             0  rcvin/s      0
file-sz      267  xmtin/s      0
ov             0  mdmin/s      0
lock-sz        7                  Press 'Ctrl c' to return to the previous frame
```

Figure 9: System activity reporting with sysadm (example)

In the following example the [CMD-MENU] function key is used to call up the command menu. By pressing the letter *e* for "exit" and the [RETURN] key you can quit the *sysadm* interface.

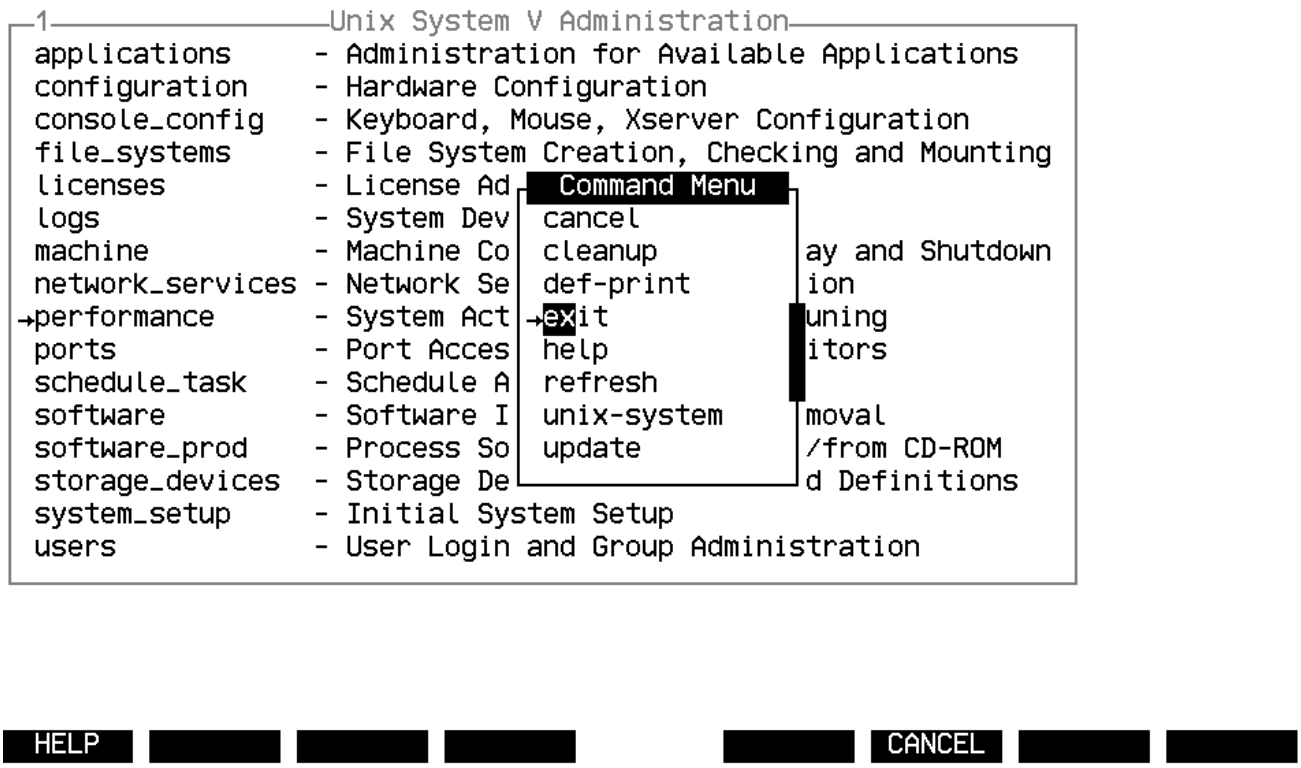

Figure 10: Typical sysadm command menu

4.3 SINIX/windows

The SINIX/windows User Environment provides you with a working environment which allows you to access graphical applications on your monitor using a mouse or the keyboard. Among other things, SINIX/windows features system administration desktools which allow you to handle system file management, login administration, and device, network and software administration using a graphical interface (also refer to the notes on SINIX/windows in section "Working environment").
SINIX/windows mouse and window operation complies with the standard conventions. For this reason it is not described any further at this point.
When you log in to the desktop for the first time, the session manager generates the user start session using the system default values. Both the system defaults and the individual session settings can be modified.

Note:

For a detailed description of the SINIX/windows configuration files see the manual "SINIX/windows User Environment V3.0, Advanced User and System Administrator Guide (CDE)" "[21]".

You can open the "Application Manager" by clicking on its icon in the main display at the bottom of the SINIX/windows window.

Figure 11: The SINIX/windows Application

The Application Manager window contains the icon for the system administration applications.

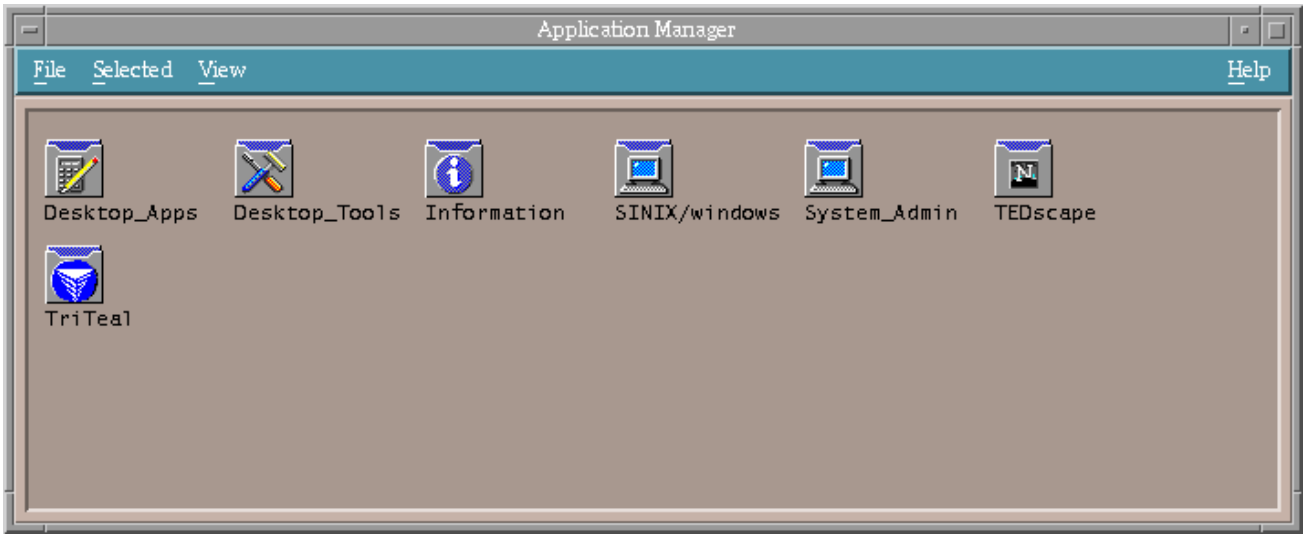

Figure 12: The SINIX/windows Application manager

Here you can access the system administration tools with SINIX/windows.

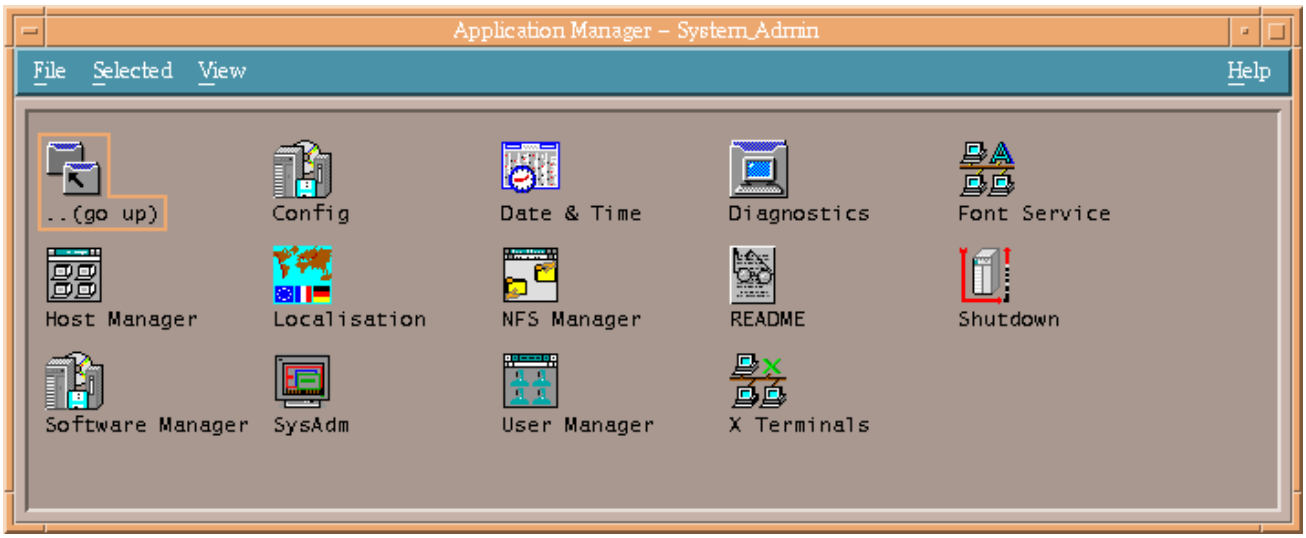

Figure 13: The system administration tools with SINIX/windows

If the *SIconf* package is installed, SINIX/windows allows you to access the graphical configuration tool *xconfig*. For further information refer to the manual "Hardware Configuration Using Config under SINIX/windows" "[11]".

4.4 The shell

The shell (also known as the command interpreter) is the controlling process under which every user executes commands. It is the user's interface to the kernel (also see section "Format and function of the /etc/passwd file"). In Reliant UNIX 5.4x the following shells are available for the system administrator to assign to user accounts:

Shell	Key
sh	Default shell, also known as the Bourne shell.
jsh	The job shell is an extension of the Bourne shell, with additional support for control and monitoring of processes (jobs) started from the Bourne shell
csh	The C shell is an extension of the Bourne and job shells with facilities for command-line editing and for accessing previously entered commands (history mechanism)
ksh	The Korn shell is an extension of the Bourne and job shells with facilities for command-line editing and for accessing previously entered commands (history mechanism)

Table 16: The Reliant UNIX 5.4x shells

Note:

The */etc/shells* file keeps a record of the shells which are cleared for use on a system. As system administrator you can define allowed shells by adding/removing the corresponding entries from the */etc/shells* file.

The C shell and the Korn shell offer more extensive functionality than the Bourne shell. The C shell is often used by C programmers because it features C-like command syntax. The Korn shell has a powerful history mechanism which is accessed using *vi* commands. Both the C shell and the Korn shell support command aliasing. However, inexperienced system administrators in particular should take great care when working with the Korn or C shell. The extended functionality also entails extra risks: for example, unchecked or undocumented definition of aliases may have unintended results.
Each user's login shell is defined in the */etc/passwd* file. You can modify the definition using the *sysadm* interface (menu items *users* - *modify* - *user*) or the *usermod* command. See section "Modifying and deleting accounts and user groups".

4.4.1 The Bourne shell

The Bourne shell is the original UNIX shell. In addition to the primary command interpreter functions of a shell, the Bourne shell includes the essential elements of a programming language.

Note:

There is a full description of the Bourne shell in the manual "Commands" "[5]", *sh* command description, and in the "User's Guide" "[6]".

4.4.2 The Korn shell

In terms of syntax and semantics, the Korn shell is largely compatible with the Bourne shell. Bourne shell scripts will run under the Korn shell. The Korn shell has a more extensive functionality than the Bourne shell; for example, it features command aliasing, more powerful shell built-ins, including some additional ones, and the history mechanism referred to above.

Note:

For detailed information on the Korn shell refer to the manual "Commands" "[5]", *ksh* command description.

A Korn shell is set up and configured in the *$HOME/.profile* file. The configuration can be extended in *$HOME/.kshrc*.
To be able to use the Korn shell you must take the following actions:

- If necessary, change the login shell in the */etc/passwd* file.
- Customize your *$HOME/.profile* file by setting environment variables. *$HOME/.profile* is also where you define aliases. Aliases are generally short replacements for complex or lengthy commands.
  In the following example the letters *ll* are defined as an alias for *ls -laF | more*.

```
# cat .profile
...
EDITOR=vi
FCEDIT=vi
HISTSIZE=10000
export EDITOR FCEDIT HISTSIZE ...
alias l='ls -laF'
alias ll='ls -laF|more'
...
#
```

If you log in again or execute the *$HOME/.profile* file, all the commands you enter will now be stored in the *$HOME/.sh_history* file. You can use *vi* commands to access the contents of this file. The following are the most important commands:

- key combination **[ESC] [K]**: retrieve the last command. You can then position yourself in the command line with the keys l, w and h and edit the command line with the keystroke commands x, cw, dw, r, R, i, a and A.
- key combination **[ESC] [/]** *pattern* **[RETURN]**: search for *pattern* in the history file, continue searching with **[/]** and **[RETURN]**.

Having edited a command line you can terminate it at any point in the text and execute it by pressing the **[RETURN]** key.
There are also a series of very useful shell built-ins, as described in the Korn shell command description (see "Commands" "[5]"). Some examples are given below:

```
# type imaker
imaker is /opt/fmas3.0/bin/imaker
```

```
# find . -name kurs.icp -print&
[1] 18513
# jobs
[1] + Running find . -name kurs.icp -print&
[2] + Done sleep 30&
# r
jobs
[1] 18513
[1] + Done find . -name kurs.icp -print&
# pwd
/var/spool/spooler
# cd /home/qm235/martin/frame/texts
# pwd
/home/qm235/martin/frame/texts
# cd -
# pwd
/var/spool/spooler
# cd -
# pwd
# /home/qm235/martin/frame/texts
```

The key combination **[CTRL] [Z]** and the *bg* command suspend an active process and set it to run in the background.

```
# sleep 1000
#[CTRL][z] pressed
[1] + Stopped sleep 1000
# bg
# jobs
[1] + Running sleep 1000&
# kill %1
[1] + Terminated sleep 1000&
#
```

4.4.3 The C shell

Like the Korn shell, the C shell also has a more extensive functionality than the Bourne shell, featuring history substitution, additional shell built-ins, command aliasing, job control and so on. The syntax of the C shell programming language is very similar to that of the C programming language

Note:

For detailed information on the C shell refer to the manual "Commands" "[5]", *csh* command description.

To configure a C shell you use the *$HOME/.cshrc* file. This file is executed whenever a C shell is started up. If the C shell is invoked as a login shell, execution of the *$HOME/.cshrc* file is followed by execution of the *$HOME/.login* file, which defines the environment and the terminal type. When a login C shell is terminated, the *$HOME/.logout* file is executed. You can customize this file as appropriate to your own needs.
The C shell, too, has a command aliasing mechanism. In the following example the alias *dir* is defined as a replacement for *ls -al | pg -e*:

```
% alias dir 'ls -al | pg -e
```

History commands allow you to access the command lines in the history list. The shell searches through the history list backwards from the most recent command line to the first. If it finds the required command line, it selects either the entire line or certain parts of it and modifies it as required.
History substitution operates on the basis of "event designators" (which define the required command line), "word designators" (which select one or more words from the defined command line) and "modifiers" (which define the changes to make to the defined command line).
In the examples below, the history list is first scanned for the most recent command line beginning with *l* (the event designator is *!l*). Then the *history* command displays the entire history list, and the history command *!60* executes command number 60 on the history list (i.e. *echo $HOME*). Finally the contents of files *file1* and *file2* are displayed, and then the event designator *!cat* searches for the most recent command line beginning with *cat*, and the modifier *s/file1/file3* replaces the string *file1* with *file3* . The event designator and the parameter are separated by a colon (:).

```
% !l
ll
total 350
drwx--x--x 3 ms     si511     1024  Sep 12 09:01 DOCUMENTS
drwx--x--x 2 ms     si511      512  Feb 10 13:43 SAG
drwx--x--x 2 ms     si511     1024  Jul 30 11:55 FM
drwx--x--x 2 ms     si511      512  Jan 21 09:14 KH
-rw-r--r-- 2 ms     si511      928  Nov 22 16:24 mbox
...
% history
...
59 ps -ef
60 echo $HOME
61 ll
% !60
% /home/martun
% cat file1 file2
Contents of file file1
Contents of file file2
% !cat:s/file1/file3/
cat file3 file2
Contents of file file3
Contents of file file2
```

Like the Korn shell, the C shell offers some very useful built-in commands. The following paragraphs briefly summarize a selection of these commands.
The *limit* command assigns limits on resource use to the current process and any processes it spawns; the *unlimit* command cancels these restrictions. In the following example the size of new files is limited to 1024K.

```
% limit
cputime         unlimited
filesize        unlimited
datasize        16384 kbytes
stacksize       16384 kbytes
coredumpsize    1024 kbytes
memoryuse       0 kbytes
descriptors     64
```

```
% limit filesize 1024
cputime         unlimited
filesize        1024 kbytes
datasize        16384 kbytes
stacksize       16384 kbytes
coredumpsize    1024 kbytes
memoryuse       0 kbytes
```

Using a feature known as the "directory stack" and the *pushd* command you can change directories without explicitly specifying the directory name. If you change to a directory using *pushd directory-name*, the new directory is added to the top of the stack. The *dirs* command lists the contents of the stack, with the topmost directory on the left. A tilde (~) stands for the home directory. *pushd* on its own accesses the topmost element on the stack, *pushd +n* accesses the *n*th element (the elements are numbered consecutively from left to right starting at 0).

```
% pwd
/home/martun
% dirs
~
% pushd /etc
/etc ~
% pushd /home/martun/KH
~/KH /etc ~
% pushd
/etc ~/KH ~
% pwd
/etc
% pushd +2
~ /etc ~/KH
% pwd
/home/martun
```

The key combination **[CTRL] [Z]** and the *bg* command together suspend a running process and move it into the background; the *fg* command moves a background process into the foreground.

```
% sleep 1000
#[CTRL][z] entered
Stopped
% jobs
[1]  + Stopped            sleep 1000
% bg %1
[1]    sleep 1000 &
% jobs
[1]  + Running            sleep 1000
& fg %1
sleep 1000
```

4.5 Virtual terminals

Anyone who has used a graphical interface is accustomed to switching between a number of windows which are open at the same time by clicking a mouse button. There is something similar for alphanumeric terminals: virtual terminals. This feature allows you to have a number of shells running on a physical terminal and switch between these shells. The shell you are currently using takes up the whole screen.
Virtual terminal functionality is based on the "shell layer" mechanism. The *shl* command activates the associated management program. For further information refer to the manual "Commands" "[5]", *shl* command description. A related utility is the management program *vtlmgr* (virtual terminal layer manager), which will only run on the system console but is considerably more useful for system administration tasks on RM200/RM400 systems.
To run the virtual terminal layer manager you use the *vtlmgr* command. Alternatively you can use function key combinations: **[ALT]-[SYSRQ] [F1]** starts up the first virtual terminal, **[ALT] -[SYSRQ] [F2]** opens a second terminal, and so on.
To switch between virtual terminals you likewise use the key combination **[ALT]-[SYSRQ]** *function-key*. In addition, **[ALT]- [SYSRQ] [H]** switches to the system console, **[ALT]-[SYSRQ] [N]** selects the next virtual terminal and **[ALT] -[SYSRQ] [P]** the previous virtual terminal. Thus it is possible, for example, to run a graphical interface in one shell layer and a DOS emulation in the next and to work on Reliant UNIX system level in a third layer. Bear in mind, though, that the more this functionality is used, the greater the system load.
To close the shell layer for a virtual terminal you use the *exit* command or the key combination **[CTRL]-[D]**, as when you exit a normal shell. The vtlmgr command is described in full in the "System Administrator's Reference Manual" "[7]".

```
# vtlmgr
# newvt
vt 1> newvt
vt 2> pwd
/
vt2> exit
vt1> exit
#
```

Note:

The *vtlmgr* command is available only if *SIgraphics* is installed. This is done automatically if VGA-Graphic or VGA-ANSI is selected as terminal type.

4.6 Useful commands and scripts

The commands and scripts discussed in this section are merely a small selection from the system administrator's toolbox. Under Reliant UNIX there is a command or script for nearly every application. Furthermore, the shell can be used as a programming language, allowing you to write other scripts or combine commands. However, the commands dealt with below form an adequate foundation set for those new to Reliant UNIX system administration.

4.6.1 df and du

One of the system administrator's chief duties is to monitor free disk space. The rule of thumb is that for a file system to function efficiently, the disk space allocated to it must never be more than 90% full. A full file system may bring the computer to a halt (though warnings are displayed on the console beforehand, e.g. *notice: /home file system full*).
The disk space monitoring commands are *du* and *df*. The *du* command is usually applied on a per-user basis, to find out which users are taking up more than the average amount of space (also refer to section "Defining user disk space quotas"). The *df* command is used to check how much space is occupied by the file systems.
In the first example below, the *df* command is used with its *-k* option. *df -k* returns the disk space occupancy of each file system in kilobytes and as a percentage. It also shows the file system mount points and the special files associated with the file systems. (Another useful option is *-b* . This simply shows the free disk space in kilobytes.) The example clearly indicates that the system administrator needs to take action, as the file system on partition */dev/ios0/sdisk000s3* (mount point */usr*) is already too full. The system administrator must make more space available as soon as possible or delete unneeded files.
The second example demonstrates a special feature. The *df* command can also be applied to file systems which are not mounted. However, the file system type must support the functionality of the command.

```
# df -k
filesystem              kbytes     used    avail capacity mounted on
/dev/root                70164    19648    43498    31%   /
/proc                        0        0        0     0%   /proc
/dev/fd                      0        0        0     0%   /dev/fd
/dev/ios0/sdisk000s2    200106    84012    68491    47%   /opt
/dev/ios0/sdisk000s3     57755    51210      769    99%   /usr
/dev/ios0/sdisk000s4     99900    37014   185356    41%   /var
/dev/ios0/sdisk000s5    333872    45052     1678    15%   /home
#
# umount /home
# df -F ufs -k /dev/ios0/sdisk000s5
filesystem             kbytes     used     avail capacity mounted on
/dev/ios0/sdisk000s5   333872    45052    255432    15%
# mount /home
# df -k /home
filesystem             kbytes     used     avail capacity mounted on
/dev/ios0/sdisk000s5    333872    45052    255432    15%   /home
#
```

In the next example, the *du* command shows the space occupied by directories in 512-byte blocks. User *pt* is clearly the guilty party. If disk space is running out, you need to investigate the matter and have a word with this user. It is also possible to set a per-user space restriction to prevent such problems arising (see section "Defining user disk space quotas").

```
# du -s /home/*
32      /home/admin
2       /home/lost+found
62      /home/oasys
51696   /home/pt
3298    /home/ra
7826    /home/sm
1092    /home/tb
2       /home/tele
702     /home/vmsys
#
```

Note:

For a full description of the *df* command refer to the "System Administrator's Reference Manual" "[7]" The *du* command is documented in the manual "Commands" "[5]".

You can check up on disk usage and file sizes using the sysadm interface, under the *diskuse* and *filesize* items on the *file_systems* menu:

sysadm interface		
Menu item	**Function**	**Shell command**
file_systems - diskuse	Report free disk space	df
file_systems - filesize	List files by size	du

Table 17: Checking disk occupancy and file sizes with sysadm

4.6.2 ps

You can use the *ps* command to view information relating to all the processes currently running. The information includes the process owner, the process ID, the parent process ID and the associated command name (also refer to section "Processes"). The source of this information is the */proc* or */dev/kmem* command.
The command output represents a snapshot of the status of processes running on the system at one time. That means that processes spawned after you call the command are not listed.
You should monitor the running process regularly. A runaway process may consume CPU time unnecessarily.
The command is mostly used when processes need to be killed. To kill a process you need to find out its process ID. You must take great care in doing so, to ensure that you do not inadvertently kill the wrong process, with potentially disastrous consequences (on killing processes refer to section "kill and killall").
The following example is a typical *ps* report.

```
# ps -ef
UID   PID  PPID  C    STIME TTY      TIME COMD
root     0     0  0 12:37:26 ?        0:10 sched
root     1     0  0 12:37:26 ?        0:00 /sbin/init
```

```
root     2     0  0 12:37:26 ?         0:00 pageout
root     3     0  0 12:37:26 ?         0:00 fsflush
root     4     0  0 12:37:26 ?         0:00 kmdaemon
root     5     0  0 12:37:26 ?         0:00 mr_daemon
root     6     0  0 12:37:26 ?         0:00 stidaemon
root   556     1  0 12:38:17 ?         0:00 /usr/lib/saf/sac -t 300
root   557     1  0 12:38:18 console   0:00 /usr/lib/saf/ttymon -g -p
Console  Login:  -d /dev/ttylc -l console
root    88     1  0 12:37:38 ?         0:00 /sbin/errord
root   189     1  0 12:37:48 ?         0:00 /usr/sbin/silsd
root   128     1  0 12:37:44 ?         0:00 cat /dev/osm
root   215     1  0 12:37:49 ?         0:00 /opt/log3/bin/log3svr -et
sdm1440
root   406     1  0 12:38:07 ?         0:00 snmpd
root   216   215  0 12:37:49 ?         0:00 /opt/log3/bin/log3tlog2
root   217   215  0 12:37:49 ?         0:00 /opt/log3/bin/log3syslogd
...
```

The amount of detail reported by the *ps* command depends on the options you use. The following table summarizes the information given under the most important column headings in the *ps* command report:

Heading	**Meaning**
UID	Login name or user ID of the process owner
PID	Process ID, unique within the system
PPID	Process ID of the parent process
C	Processor utilization
STIME	Time when the process was spawned. For the first 24 hours the time of day is reported, thereafter the date.
TTY	Terminal from which the process was spawned. A query (?) means that there is no controlling terminal (daemon process or zombie).
TIME	Cumulative execution time for the process
COMD	Command associated with the process

Table 18: Principal headings in the ps command report

Note:

For a full description of the *ps* command refer to the manual "Commands" "[5]".

You can check up on current processes using the sysadm interface, under *list_all* on the *processes* submenu of the *performance* menu:

sysadm interface		

Menu item	Function	Shell command
performance - processes - list_all	List processes	ps

Table 19: Process reporting with sysadm

4.6.3 whodo, finger and fuser

You can use the *whodo* and *finger* commands to check up on the activities and processes of users who are currently logged in. The *fuser* command reports on the processes of users accessing a particular file system. Especially when the system load is unusually high, you should check whether one user's processes are responsible. It may be possible for the user to have work carried out at night with the aid of the *at* command (see section "Task scheduling").
The *whodo* command reports the date, operating system information, terminal parameters, login names, process IDs, CPU times and active commands. If you specify a login name on the command line, the report is restricted to the associated user. The sources of the information are the files */etc/passwd* , */var/adm/utmp*, */etc/ps_data* and */proc/process-ID*.

```
# whodo
Wed Oct 1 10:11:25 1997
Reliant UNIX-N
console      root      7:19
console        557    0:00 sh
console        714    0:00 whodo
inet/26      arthur   12:39
inet/26        564    0:00 ksh
#
```

The *finger* command also reports information relating to users. In its simplest form it does not report in as much detail as *whodo* , but it has the advantage that it is not restricted to local use. It can also access remote computers over a network. For details refer to the manual "Network Administration" "[8]".

```
# finger
Login       Name              TTY          Idle    When    Where
arthur   Arthur Dent          inet/26         2 Thu 13:04  si511eddie
root     0000-Admin(0000)     pts/2           1 Thu 13:10
#
# finger -l
Login name: arthur                    In real life: Arthur Dent
Directory: /home/arthur               Shell: /sbin/ksh
On since Jan 4 07:19:45 on inet/26    from si511eddie
1 hour 48 minutes Idle Time
No unread mail
No Plan.
Login name: root                   In real life: 0000-Admin(0000)
Directory: /
On since Jan 4 09:39:56 on console
12 seconds Idle Time
No unread mail
No Plan.
#
```

The *fuser* command reports which processes are using resources on a file system. *fuser -u* lists the login names as well as the processes. As with the *ps* command the information is a snapshot, which means that processes spawned after you call the command are not listed In the first example below, the command lists all the user processes which are accessing the file system on partition */dev/ios0/sdisk000s5*, i.e. the file system mounted on */home*.
In the second example, *fuser -k* invokes the *kill -9* command to kill all processes which are accessing the file system on partition 5 (*/home*). Also refer to section "kill and killall").

Warning:

fuser -k should be used only in emergencies, such as when the file system is overflowing, or before a backup when there are no users logged in but there are still processes running on the file system being backed up.

```
# fuser -u /dev/ios0/sdisk000s5
/dev/ios0/sdisk000s5: 22364c(ms) 22359co(arthur) 22136c(ms) 2212 7c(ms)
22124co(ms) 22123co(ms) 22122c(ms) 22121c(ms) 22118c(ms) 22117co(ms)
21775c(arthur)
#
# fuser -k /dev/ios0/sdisk000s5
#
```

Note:

For a full description of the *whodo* and *fuser* commands refer to the "System Administrator's Reference Manual" "[7]". The finger command is documented in the manual "Commands" "[5]".

4.6.4 cleanup

cleanup is a shell script used to delete garbage files such as unwanted core dump and log files. The script resides in the */sbin* directory and can be customized by the system administrator. There is usually a call to this script included in the file */var/spool/cron/crontabs/root* . If this is not the case, you should add one or run the script regularly "by hand" (see also section "The cron service").

```
# cleanup &
#
```

4.6.5 nohup

Processes sent into the background by means of the *&* character (*cleanup &*, for example) are terminated when the calling user logs out, because such processes are associated with the login shell. This behavior is not always desirable. By using the *nohup* command you can send commands or shell scripts into the background such that they are not terminated when you log out.

```
# nohup cleanup &
34562
Sending output to nohup.out
#
```

Note:

For a full description of the *nohup* command refer to the manual "Commands" "[5]".

4.6.6 kill and killall

The *kill* command sends signals to processes and is mostly used to kill (terminate) processes. Nonprivileged users can only ever kill their own processes. As system administrator you can kill almost any process; the only process you cannot kill is the process with process ID 1, the *init* process, because the operating system must always be shut down "gracefully", for example with */sbin/shutdown* (see section "Switching run levels"). Before you can kill a process, you need to find out the process ID using the *ps* command (see section "ps"). You must take great care in doing so, to ensure that you do not inadvertently kill the wrong process, with potentially disastrous consequences for the entire operating system. If you do make a mistake of this kind, it may be necessary to reboot the operating system.
In the following example the subshell *-sh* of user *ms* is killed.

```
# ps -u ms
PID TTY TIME COMD
25371 12  0:14 -ksh
26023 12  0:10 m
26233 12  0:00 -sh
#
# kill -9 26233
# ps -u ms
PID TTY TIME COMD
25371 12  0:14 -ksh
26023 12  0:10 m
#
```

The *killall* command, which is included in the *shutdown* script, kills all processes which are accessing file systems, thereby allowing the file system to be unmounted.

Warning:

The *killall* command should be used only in single-user mode.

Note:

For a full description of the *kill* command refer to the manual "Commands" "[5]". The *killall* command is documented in the "System Administrator's Reference Manual" "[7]".

You can kill processes using the sysadm interface, under *signal* on the *processes* submenu of the *performance* menu:

sysadm interface		
Menu item	**Function**	**Shell command**
performance - processes - signal	Killing processes	kill

Table 20: Killing processes with sysadm

4.6.7 last

To monitor the login and logout times of your users you use the *last* command. *last* evaluates the information in the */var/adm/utmp* file (see section "Log files"). As remote logins are logged as well, *last* is useful as a means of identifying security loopholes.
The output from *last* may be very extensive. If the */var/adm/utmp* file is not regularly cleared (using the *cleanup* script, for example), all logins since the last time the operating system was booted are recorded there. Therefore it is best to redirect the output to the *pg* or *more* command.

```
# last | pg
mn     pts/3       othello    Thu Mar 19 13:14 still logged in
mn     pts/3       othello    Thu Mar 19 13:13 - 13:14 (00:00)
.rlogin /dev/pts/3            Thu Mar 19 13:13 still logged in
root   pts/2                  Thu Mar 19 13:10 still logged in
mi     term/tty004            Thu Mar 19 13:04 still logged in
getty  term/tty011            Thu Mar 19 12:48 still logged in
bt     term/tty011            Thu Mar 19 12:48 - 12:48 (00:00)
bt     term/tty011            Thu Mar 19 12:40 - 12:48 (00:07)
getty  term/tty011            Thu Mar 19 12:39 - 12:40 (00:01)
bt     term/tty011            Thu Mar 19 12:39 - 12:39 (00:00)
getty  term/tty007            Thu Mar 19 12:36 still logged in
st     term/tty007            Thu Mar 19 12:36 - 12:36 (00:00)
st     term/tty007            Thu Mar 19 12:26 - 12:36 (00:10)
...
#
```

Note:

The *last* command is described in full in the manual "Commands" "[5]".

4.6.8 su

The *su* command changes the user ID you are working under, thereby switching you temporarily to another account. The user with user ID 0, the superuser or system administrator, can do this without having to supply the associated password.
The command name *su* is often mistakenly held to mean "become **s**uper**u**ser", because the command is interpreted as *su root* when called without command-line arguments.

Warning:

On security grounds the system administrator should avoid running the *su* command on another user's terminal. It is fairly simple for an experienced user to secretly log the password check and thus find out the system administrator's password.

A less well-known feature of the *su* command is illustrated in the following example. If you use a dash as a command-line argument, the *$HOME/.profile* file of the specified user, here *root*, is evaluated as well.

```
$ id
```

```
uid=421(murtan) gid=231(si511)
$
$ su - root
Password:
# id
uid=0(root) gid=1(other)
#
```

Note:

For a full description of the *su* command refer to the manual "Commands" "[5]".

4.6.9 find

One of the most important and powerful Reliant UNIX commands is *find*. You should devote time to closely studying the command description and experimenting with the command. The following examples illustrate the major functions of *find* .
Listing all the directories and files which belong to the *root* file system:

```
# pwd
/
# find . -mount -print
/lost+found
/etc/cron.d
...
#
```

Deleting all files named *core* or with the extension .*bak* which are more than seven days old and are below the */u* directory (recursively).

```
# find /u \( -name core -o -name "*.bak" \) -atime +7 -exec rm {} \;
...
#
```

Recursively copying all directories and files below the */home/ms* directory to cartridge tape.

```
# cd /home/ms
# find . -depth -print | cpio -ocvB > /dev/rmt/c0s0
...
#
```

Interactively deleting all files more than 2000 blocks in size anywhere below the current position in the directory hierarchy.

```
# find . -depth -type f -size +2000 -ok rm {} \;
< rm ... ./home/as/nonsens >?   y
#
```

Detecting security loopholes by checking the permissions of executable commands and scripts which have the s bit (set-user-ID bit) set.

```
# find / -perm 4000 -user root -print
...
#
```

Note:

For a full description of the *find* command refer to the manual "Commands" "[5]".

4.6.10 file

The *file* command supplies you with information about the type of a file. This may be important when you want to edit a file, because not all types of file can be read into an editor or listed with the *cat* , *more* and *pg* commands.
The command checks the file type by examining the information in the */etc/magic* file. If extra application software such as FrameMaker has been installed, special file types can be identified only if the */etc/magic* file is updated after installation.

```
# file *
dvz1:           directory
test:           ascii text
alt2neu:        commands text
whodo:          ELF 32-bit LSB executable 80386 Version 1
ebtpriv:        directory
fifo:           fifo
...
#
```

Note:

For a full description of the *file* command refer to the manual "Commands" "[5]".

4.6.11 grep

The *grep* command can be used to filter specified patterns from ASCII files or standard input. The patterns can be given in the form of regular expressions. The related *egrep* command is more suitable for complex patterns or extended regular expressions.
The examples below illustrate a number of applications for the *grep* command. In the first two examples the output of the *ls* command is restricted to lines matching the specified pattern. In the third example, only lines matching the pattern are extracted from a file. The last example takes the output of the *ps* command and first extracts the lines containing the letters *ms*. As the command *grep ms* itself appears in this output, the third part of the command line removes all lines containing the pattern *grep* (you can achieve the same result by using appropriate options of the *ps* command).

```
# ls -l | grep "Aug 19"
-rwx--x--x 2 martin si511    77656  Aug 19 18:59 test
#
# ls -l | grep "^d"
drwx--x--x  2 ms     si511       512  Feb 10 13:43 TEST
#
# grep hamlet /etc/hosts
139.101.122.100 hamlet
#
# ps -ef | grep ms | grep -v grep
ms 25371 12  0:14 -ksh
#
```

Note:

For a full description of the *grep* command refer to the manual "Commands" "[5]".

4.6.12 compress and uncompress

The *compress* command is a file compression utility. Compression is a short-term alternative to off-line file storage on floppy disk or tape.
The *pack* command does a similar job but generally does not produce such a good compression ratio.

```
# ls -l
total 2
-rwx--x--x  2 ms     si511    2990656  Aug 19  test.mif
# compress test.mif
# ls -l
total 2
-rwx--x--x  2 ms     si511    1478783  Aug 19  test.mif.Z
#
# uncompress test.mif.Z
# ls -l
total 2
-rwx--x--x  2 m      si511    2990656  Aug 19  test.mif
#
```

Note:

For a full description of the *compress* command refer to the manual "Commands" "[5]".

4.6.13 script

The *script* command records everything displayed on the terminal during a session and logs it in the specified file. In the following example it is used to log the installation of a software package.

```
# script /home/logs/installprot1
Script started on Tue Aug  9 16:29:20 1994
# pkgadd -d /dev/ios/stape003
...
...
# exit
script done on Tue Aug  9 16:37:00 1994
#
```

Note:

For a full description of the *script* command refer to the manual "Commands" "[5]".

4.6.14 Other useful commands

The *strings* command scans files for ASCII strings and writes them to standard output. This allows you to find strings in files which cannot be read into an editor or listed with the *cat*, *more* and *pg* commands. The following example extracts the readable strings from the */var/crash/dump.0* file.

```
# strings /var/crash/dump.0 | pg
dumpsyshdr
@etS0
```

```
004A
HP@
@ !
@@A
`
`aAj
ppprwqpp
'{@x
'{@x
menue
DRIVE
y(0)
tty(0)
lantype=ethernet
ethernetaddr=0:0:e4:3:9:72
...
#
```

Note:

For a full description of the *strings* command refer to the manual "Commands" "[5]".

The *ksh -xv* command is useful for debugging your own shell scripts or for examining those of other users. The script is executed in a subshell and the results are redirected to a file.
The following example logs the execution and output of the *cleanup* script in the file */tmp/logs* (on *cleanup* refer to section "cleanup").

```
# ksh -xv /sbin/cleanup 2>/tmp/log&
#
```

You can make life as system administrator easier for yourself by using the shell's programming language capability. The example below is intended to whet your appetite. Bourne shell programming is fully described in the manual "Commands" "[5]" and in the "User's Guide" "[8]", so the example here is kept very short.
The following shell script uses a *for* loop to copy to a DOS floppy disk all files with names beginning with an uppercase letter from *A* to *K* and ending in *.mif*.

```
# for i in ls [A-K]*.mif
>do
>doscp $i a:$i
>done&
#
```

The *truss* command executes a command you specify and produces a trace of the system calls it performs, the signals it receives and the machine faults it causes.
Even without any special programming skills you can see from the following example that: the shell-level command *ls -l* is called by its absolute path name (*/sbin/ls*); the language variable is evaluated; the time is recorded; entries in the */etc/passwd* and */etc/group* files are checked; and finally the output is prepared.

```
# truss -o /tmp/logfile ls -l /stand
# more /tmp/logfile
execve("/sbin/ls", 0x7FFFEB90, 0x7FFFEBA0)  argc = 3
open("/usr/lib/locale/En_US.ASCII/LC_CTYPE", O_RDONLY, 01011) = 4
read(4, "\0                    ( ("..., 521)      = 521
close(4)                                          = 0
```

```
open("/usr/lib/locale/En_US.ASCII/LC_NUMERIC", O_RDONLY, 02) = 4
read(4, " . ,", 2)                                  = 2
close(4)                                            = 0
time()                                              = 789916492
ioctl(1, TCGETA, 0x7FFFE8EC)                        = 0
open("/etc/passwd", O_RDONLY, 0666)                 = 4
open("/etc/group", O_RDONLY, 0666)                  = 5
brk(0x0045F4B0)                                     = 0
brk(0x004674B0)                                     = 0
lxstat(2, "/stand", 0x7FFFE878)                     = 0
open("/stand", O_RDONLY, 01)                        = 6
fcntl(6, F_SETFD, 0x00000001)                       = 0
fxstat(2, 6, 0x7FFFE808)                            = 0
brk(0x004684B0)                                     = 0
getdents(6, 0x00467FF0, 1048)                       = 64
lxstat(2, "/stand/unix", 0x7FFFE810)                = 0
lxstat(2, "/stand/unix.old", 0x7FFFE810)            = 0
getdents(6, 0x00467FF0, 1048)                       = 0
fxstat(2, 6, 0x7FFFE810)                            = 0
close(6)                                            = 0
open("/usr/lib/locale/En_US.ASCII/LC_COLLATE", O_RDONLY, 021363770) = 6
fxstat(2, 6, 0x7FFFE714)                            = 0
read(6, "\0\0\014\0\0\0\0\0\00414".., 1044)         = 1044
close(6)                                            = 0
brk(0x004694B0)                                     = 0
open("/usr/lib/locale/En_US.ASCII/LC_MESSAGES/uxcore.abi", O_RDONLY, 01
7777764042) = 6
fxstat(2, 6, 0x7FFFE760)                            = 0
fcntl(6, F_SETFD, 0x00000001)                       = 0
mmap(0x00000000, 42268, PROT_READ, 0, 6, 0)         = 0x04801000
ioctl(1, TCGETA, 0x7FFFDE6C)                        = 0
write(1, " t o t a l   1 7 5 0 4\n", 12)           = 12
lseek(4, 0, 1)                                      = 0
...
#
```

Note:

For a full description of the *truss* command refer to the manual "Commands" "[5]".

4.7 Documentation

The manuals, too, both printed and online, count as system administration tools. This section summarizes the manuals which are of most use to the system administrator. Some of these manuals are for specific machines, which means that there are a number of versions of the manual, each tailored to a particular type of system. This distinction is not explicitly discussed in the synopsis below.
The documentation for system software is surveyed in the "Documentation Guide" "[4]".
The following manuals are recommended for system administration and general use:

- Operating Manual (machine-specific)
  The Operating Manual describes hardware installation and system startup and operation.
- Installation Guide (machine-specific)
  The Installation Guide describes initial and update installations of Reliant UNIX.
- Documentation Guide
  This manual contains an overview of the documentation for Reliant UNIX 5.44, including thumbnail sketches of the principal manuals, and an index which, via task descriptions, references the manuals with the corresponding information.
- Commands
  The two-volume "Commands" manual describes the Reliant UNIX user commands.
- User's Guide
  The "User's Guide" explains the principles of Reliant UNIX, surveys the essential elements of the operating system (such file systems and processes) and describes the chief tools that Reliant UNIX supplies (the Bourne shell, editors, awk, mail).
- System Administrator's Reference Manual
  This manual describes the commands, file formats and functions needed for system administration.
- Network Administration
  This manual describes network administration tasks for system and network administrators.
- System Administration and Hardware Configuration, SYSADM
  This manual describes the *sysadm* interface and the accompanying *Config* configuration tool. *Config* is an easy-to-use tool for I/O device configuration and management.
- Virtual Disks
  This manual describes the different types of virtual disk (including mirrored disks) and how they are configured.
- Veritas File System (VxFS) V1.2, System Administrator's Guide
  This guide is an introduction to system administration for the Veritas file system. It also discusses diagnostic and error messages and appropriate user responses.

4.8 Online documentation and online help

A collection of frequently required manuals is also available online. The online documentation is available on CD and in the World Wide Web via the Siemens Nixdorf homepage.
You can order the CD containing the online documentation from your local Siemens Nixdorf office.
The operating system shipment includes the man pages. With the man pages installed you can enter *man command-name* to call up a description of a command that interests you.
The setting of the *LANG* variable defines the language in which the man pages are displayed. German man pages are available in addition to the English pages.
The *man* command pipes its output through the paging command defined in the *PAGER* variable (the *more* command is used if *PAGER* is undefined). Thus you should know how the selected paging command works. Those using the graphical interface can also access the man pages by way of the *xman* X client.

```
# man format
format(8)                                                    format(8)
NAME
format - online disk formatter on RM600
SYNOPSIS
/sbin/format
DESCRIPTION
/sbin/format is a shell script used to format disk drives.
It requires no arguments and it prompts for all the
parameters it needs.
Supported IOS SCSI disks are self-configuring, so no
system configuration information (via setinfo or PCOS) is
necessary. Unsupported devices may be configured using
setinfo(8), see the  storage device management
documentation in the System Administrator's Guide.
Defective sectors are mapped and saved by the SCSI disk
controller.  The Pyramid Standard Disk Format (PSDF) data
structures are written onto the disk at the completion of
the format. A Format Information Table is also written on
the disk, which contains the time of last format, and
drive serial number information. Any time the Format
Information Table is printed, the SCSI defect list is read
and printed as well.
...
#
```

If you already know what a command does, but are not sure what options are available, you can simply call the command with the *-?* option. The command will not be executed, and for most commands a diagnostic known as a usage message is displayed instead. This usage message includes a list of the available options.

```
# tar -?
tar: ?: unknown option
Usage: tar -{txruc}[0-9vfbk[FX]hLiBelmnopwA] [tapefile] [blocksize]
[tapesize] [argfile] [exclude-file] [-I include-file] files ...
#
```

One other form of online help is available with the Korn shell, and that is that the history function can be used to search for commands which have already be run under the user's

login name (see section "The Korn shell").

5 System configuration and installation

This chapter deals with two very important aspects of system administration: installing and configuring the operating system. You must set aside sufficient time to plan these duties in advance, because any mistakes you make in the process take considerable time and effort to rectify later.

Note:

For further information on operating system installation refer to the Installation Guide for your system "[2]", "[3]".

5.1 Installing the operating system

Siemens Nixdorf ships its Reliant UNIX systems with the operating system preinstalled. To start up the system you simply have to switch on the power, log in as root and then immediately change the administration passwords. However, the preset installation is not always appropriate to local requirements. If this is the case, the operating system will need to be reconfigured, and reinstallation may be necessary.
If you reinstall the operating system, you must decide in advance how you want the disks to be partitioned. The primary factor in your decision-making is the hardware configuration of your system. The chief aspects of this are the number hard disks and the hard disk capacity. One other factor apart from the operating system software is the additional application software you plan to install.
As operating system installation is a machine-specific procedure, you should be sure to study the relevant Installation Guide and the operating system release notes, where all the important information is provided. The Installation Guide also includes an extremely useful worksheet where you can record your system's hardware setup and the planning data you have designed to match it. In your plans you should take at least the following aspects into account:

- Are you going to network the computer?
  If so, a LAN board must be present and all the networking components of the operating system must be installed. You should also have a host name and an Internet number which is valid or at least unique within the local area network. Your network administrator is usually responsible for allocating these parameters.
- Which hard disk is to be used as the system disk?
- How are you going to partition the hard disk?
  Your plans for hard disk partitioning should be governed by the available hardware and by the application software you will be installing later. If you have two hard disks, the file systems should be shared evenly between them. The principles of hard disk partitioning were dealt with in depth in chapter "Guided tour of the Reliant UNIX system".
  You also need to decide whether one partition should initially be left free. Some software programs, such as parts of the INFORMIX® relational database management system, for internal purposes prefer accessing data via a free partition. For such eventualities you will need to reserve a partition when you are planning your installation.
  In addition you have to determine the size of the file system which will contain the

optional software (*/opt* file system), taking into account the particular software products you plan to install on the system. If in doubt, it is best to make the file system a little on the large size.

- Are you going to configure mirror disks using *mroot*?
- Are you going to boot from a RAID array?

It may take as much as two hours to install the operating system. The installation program will help you with important decisions. For example, it offers recommendations on partition layout and size, file system types and the size of the swap area, basing the latter on the size of main memory.

Expert opinion differs as to whether large or small file systems are more beneficial. The deciding factor will in fact often be more to do with personal requirements. If you want to have an entire file system backed up at night, it is obviously easier for you if the system will fit on one streamer tape cartridge. If a DAT drive is connected, you need to consider if one file system or several file systems are to be backed up without a change of tape.

The installation procedure is only briefly outlined here; it is described in full in the relevant installation guides.

Installation begins with an interactive question-and-answer phase establishing important installation parameters (such as formatting and partitioning of the system disk, terminal type, node name and networking parameters). On completion of this phase the hard disk is partitioned and the file systems are set up.

Next the operating system software itself is installed. This software consists of a series of discrete software packages which are installed sequentially. This is in keeping with the packaging mechanism for software installation: the operating system components and the application software are all shipped and installed as packages.

The operating system can be installed from various media. The default installation medium is CD-ROM. Reliant UNIX can also be installed from a network (in which case, however, you require the installation CD on the server).

5.2 Connecting devices

After the operating system has been installed, the console terminal is as a rule the only device which is ready to use. Any other devices which are attached, such as additional terminals, printers or other hard disks, must be identified to the operating system and configured.
For each hardware connection the */dev* directory or one of its subdirectories includes a special file which allows the operating system to access the hardware. The special files you need are usually already present. However, if extra hardware is added, the special files need to be created explicitly (see section "Special files").
Once an additional device is correctly connected to the system, it still needs to be configured so that it can be accessed by the operating system. You can simplify the configuration procedure by using the sysadm interface (see section "The sysadm interface").

Note:

For in-depth information on hardware configuration using the sysadm interface refer to the manual "System Administration and Hardware Configuration, SYSADM" "[10]"

5.2.1 Configuring terminals

In earlier versions there was a separate process responsible for monitoring login requests for each terminal or terminal port. Under Reliant UNIX 5.4x this monitoring service is replaced by a single process, the Service Access Facility (SAF). This is a process which is spawned by the *init* process on the basis of the */etc/inittab* file when the operating system is booted and which in turn starts up the port monitor, */usr/lib/saf/ttymon* . Terminal connections are now implemented by configuring *ttymon*.
Monitoring is controlled by two commands: *sacadm*, which controls the port monitors, and *pmadm*, which manages the services. The port monitoring model is described in section "Port monitoring by the System Access Facility (SAF)".
To configure terminals you should always use the sysadm interface, selecting the *configuration* menu item, and you should closely follow the instructions given in the accompanying terminal documentation and in the Operating Manual for your computer.

sysadm interface		
Menu item	**Function**	**Shell command**
configuration - load	Terminal configuration	termadd

Table 21: Configuring terminals with sysadm

5.2.2 Configuring printers

If the print service software package has been installed with the operating system, the print spooler is automatically loaded by a script in the */etc/rc2.d* directory when the system goes into multiuser mode. The print service supplies powerful tools for controlling printers. Print

spooler administration is described in the manual "Xprint V5.0, User and Administrator Guide" "[15]" and "Xprint V5.0, Reference Manual" "[16]".
Reliant UNIX supports all the common printer types. However, you must read the documentation and the release notes to check that your particular printer can be used. When in doubt, consult Siemens Nixdorf customer service.
The easiest way to configure printers is to use the sysadm interface (menu items *configuration - load printers*).

sysadm interface		
Menu item	**Function**	**Shell command**
configuration - load - printers	Printer configuration	Depends on the spooler

Table 22: Configuring printers with sysadm

5.2.3 Configuring hard disks

The hard disk, too, is a device that must be identified to the operating system. Consequently, a new hard disk, or one you are upgrading, needs to be configured, using either the sysadm interface or the *dksetup* command.
The *dksetup* command initializes a new hard disk or updates an existing disk. It performs the following functions:

- it creates device nodes in the */dev* directory
- it formats the hard disk
- it initializes data structures for hard disk configuration.

Note:

For a detailed description of *dksetup* refer to the "System Administrator's Reference Manual" "[7]".

You can also configure a hard disk using the sysadm interface (menu items *configuration - load - storage_devices*.

sysadm interface		
Menu item	**Function**	**Shell command**
configuration - load - storage_devices	Configuring hard disks	dksetup, dkpart

Table 23: Configuring hard disks with sysadm

Note:

The Reliant UNIX operating system provides support for virtual disks. A virtual disk operates in the same way as a conventional physical hard disk, with a pseudo device

driver mapping all logical I/O requests to physical disks. The virtual disk mechanism allows you, among other things, to define file systems which are distributed among several physical disks, and it helps to improve performance and enhance system security.
For further information on virtual disks refer to section "RAID strategies", to section "Disk mirroring" and to the "Virtual Disks" manual "[12]" and the manual "RAIDmaster, System Administrator's Guide" "[33]".

5.3 Setting up a swap area

If the kernel is handling many processes at the same time, main memory utilization is very high. This may result in processes being moved out of main memory to disk until it is their turn to be serviced. The processes are temporarily placed in an area known as the "swap area", and the mechanism is known as "swapping". As it is not always necessary to load the entire program into RAM, but only those parts which are currently required, it possible to swap out only parts of the program. This mechanism is known as "paging".
The swap area is usually in a separate partition which is set up when the operating system is installed (see section "Hard disk partitioning"). The swap area must be at least as large as main memory; but to avoid bottlenecks and the consequent degradation of CPU performance it is best if it is twice the size of main memory. If the swap area that has been allocated proves to be too small, in extreme circumstances the system may halt and crash. If this is the case, you must set up an extra swap area or improve the configuration of the hardware, which in the long term is the better solution.
To set up an extra swap area you use the *swap -a* command. If there is no free partition available, you must either repartition the disk or create a swap file with a size which is a multiple of the 512-byte block size. Note that repartitioning is extremely time-consuming.

```
# swap -l
path                dev  swaplo blocks   free
/dev/ios0/sdisk000s1 4,1       0 262392 245808
# swap -a /dev/ios0/sdisk001s1 0 60000
#
# echo ' ' | dd of=/opt/swapfile seek=6000
# swap -a /opt/swapfile 0 6000
#
```

For a full description of the *swap* command refer to the "System Administrator's Reference Manual" "[7]".

5.4 Networking your computer

The following section contains a brief description of how to network your computer. You should contact the network administrator, as you will require network-specific information such as the Internet address.
If you make intensive use of networks, you should first study this subject in more detail. The following only describes how to network your computer using the SYSADM user interface.

Note:

For more information on networks refer to the manual "Network Administration" "[8]". Administering network services is described in the manual "System Administration and Hardware Configuration Using the SYSADM User Interface" "[10]".

Before you connect your computer to the LAN, you should ask the network administrator for the following information:
- your computer's Internet address
- your network administration computer's Internet address (if NIS is used)
- your computer's alias
- your computer's node name
- the domain name

Make sure that the network administration computer recognizes your system. (Your system can be announced to the network administration computer via the SYSADM menu items *network_services* - *lan* - *manage* - *computer* - *add*.)

· Enter the network administration computer's Internet address on your system. To do this, select the SYSADM menu items *network_services- name_to_address - inet - hosts*
· If not yet specified, enter the computer's node name. To do this, select the SYSADM menu items *system_setup - nodename - set* and enter the node name agreed on with the network administrator.
Save the specified node name with SAVE.

If NIS (Network Information Service) is not active, you must enter your system name and the Internet address in the */etc/hosts* file. To do this, select the SYSADM menu items *network_services- name_to_address - inet - hosts* or use an editor. Restart the computer.

· Configure your computer's LAN controller. Select the SYSADM menu items *configuration - load - boards - motherboard*.
An input mask is displayed in which you define the Internet address, the host name and alias, and, if you wish, the Internet mask and broadcast address.

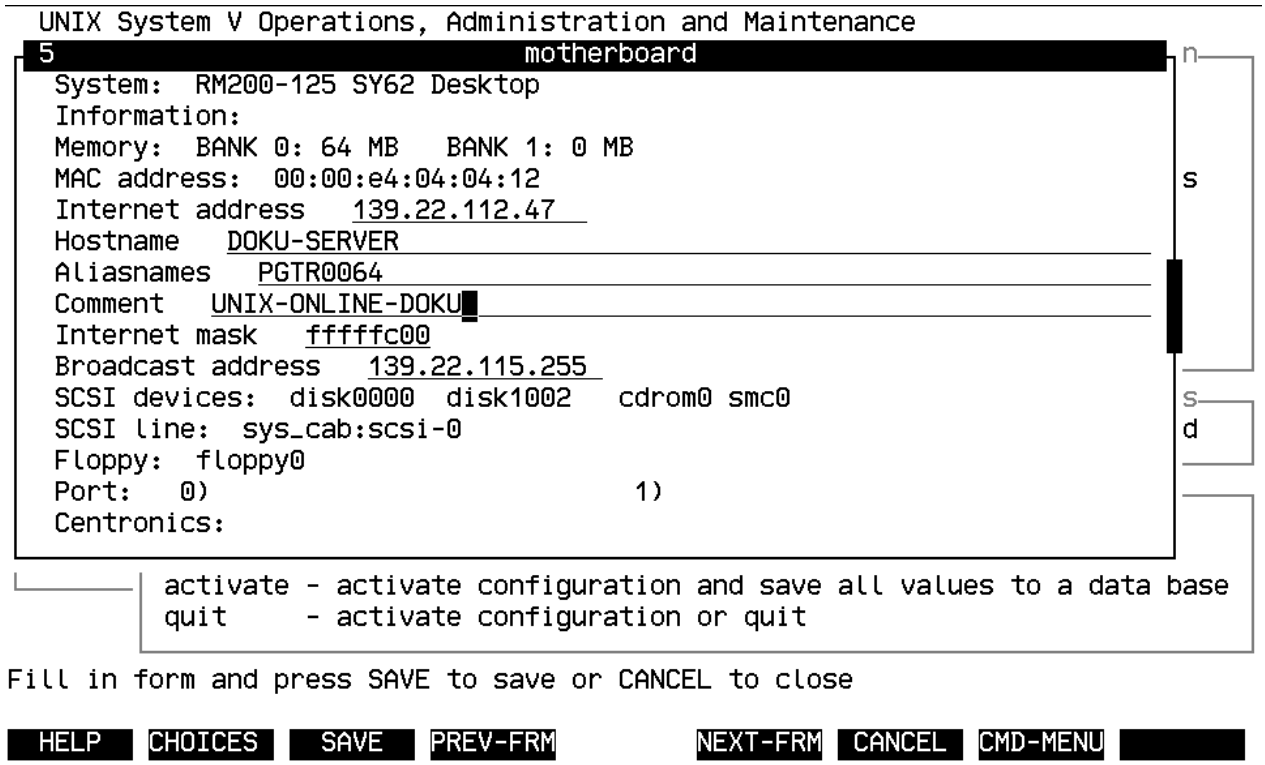

Figure 14: Configure the computer's LAN controller via SYSADM

Save the LAN controller configuration with SAVE.

- Mount the computer in the network via the menu items *network_services - lan - connect*.

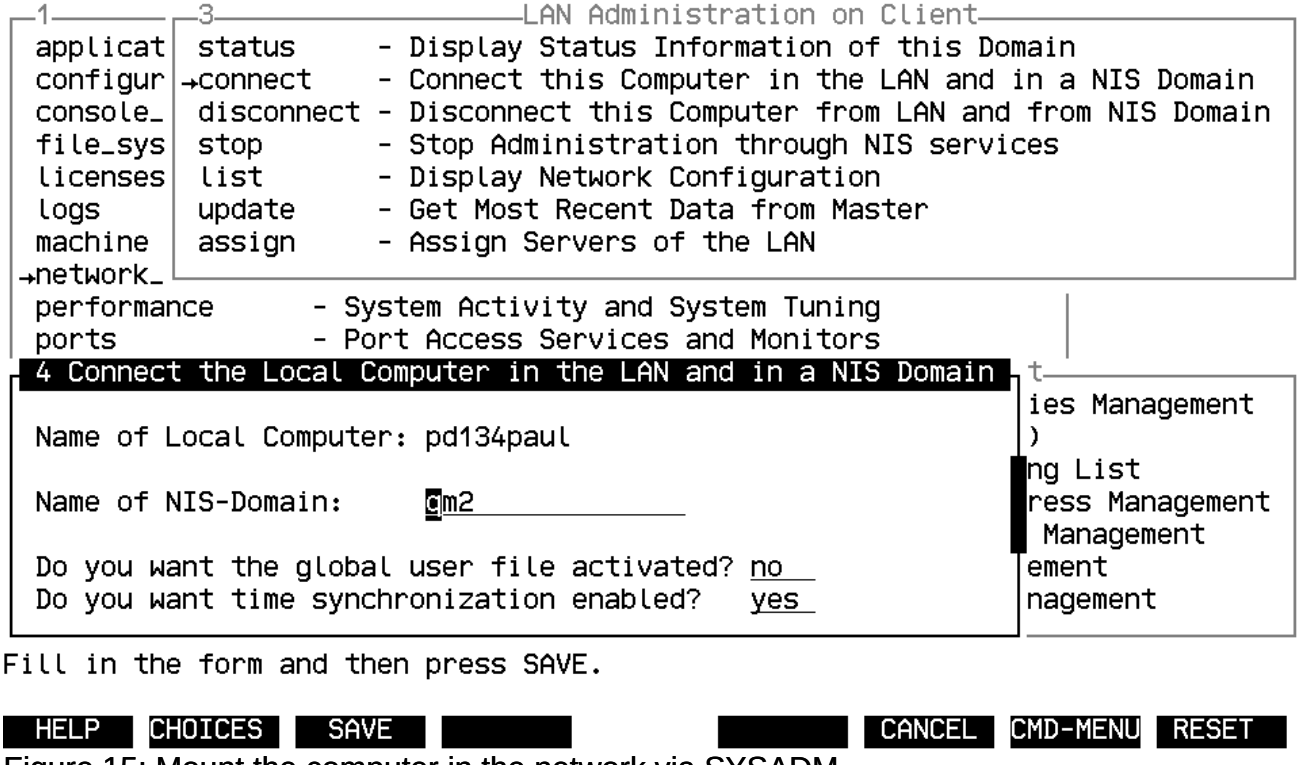

Figure 15: Mount the computer in the network via SYSADM

5.5 Customizing your system

The behavior of the operating system on your computer is controlled by configuration files. These are ASCII files used to set system variables and default values that you can customize to your own requirements. The files in the */etc/default* and */etc/skel* directories in particular offer many opportunities to tune the performance of the operating system. The following section describes the most important files and variables.

5.5.1 The /etc/default directory

The files in the */etc/default* directory contain default settings for various system components. Therefore the contents of the directory and of the files it contains may vary from machine to machine.
You should examine these files to get some idea of what you can tailor to your own needs. If you are uncertain about changes you would like to make, you should consult the relevant operating system documentation. Whatever you do, you should take great care, as the changes you make can have a major effect on the productivity of your users.

5.5.2 Files in /etc/default

This section describes the chief configuration files in the */etc/default* directory. */etc/default* may contain the following files:

```
# ls -l /etc/default
total 188
-rw-r--r--   1 root     root        2080 Jul 10 11:13 SIterms
-rw-r--r--   1 root     root          42 Jul 10 10:45 boot
-rw-r--r--   1 bin      bin           12 Jun 28 02:12 country
-rw-r--r--   1 bin      bin           59 Jul 10 10:36 cron
drwxr-xr-x   5 root     root         512 Jul 10 10:30 diskinfo
-rw-r--r--   1 root     other        126 Jul 10 10:36 dumpsave
-rw-rw-rw-   1 root     root          97 Jul 10 13:33 elbslog
drwxr-xr-x   3 root     root         512 Jul 10 13:24 floppyinfo
-rw-r--r--   1 bin      bin          686 Aug  5 09:43 inet
-rw-r--r--   1 root     other        686 Jul 24 13:33 inet.NTP
-rw-r--r--   1 root     other        421 Jul 10 13:33 inet.new
-rw-r--r--   1 root     other        344 Jul 10 13:33 inet.save
-rw-r--r--   1 root     other        420 Jul 10 13:33 inet.tmp
-rw-r--r--   1 root     other        344 Jul 10 13:33 inet.x
-r--r--r--   1 root     sys          146 Jul 10 10:36 init
drwxr-xr-x   2 root     other       3072 Jul 10 11:49 install
drwxr-xr-x   3 root     root         512 Jul 10 11:24 jukeboxinfo
-r--r--r--   1 root     other         17 Jul 10 10:44 language
-rw-r--r--   1 root     other       4961 Jun 28 03:28 log3struct
-rw-r--r--   1 bin      bin         5507 Jul 10 10:40 login
-r--r--r--   1 root     sys          668 Jul 10 11:24 msdos
-rw-r--r--   1 root     root         828 Jul 10 11:40 oedrc
-rw-r--r--   1 bin      bin          342 Jul 10 10:36 passwd
-rw-r--r--   1 bin      bin          850 Jul 10 10:36 su
-r--r--r--   1 root     bin          225 Jul 10 10:36 tapedrive
-rwxr-xr-x   1 bin      bin         1404 Jul 10 11:22 tar
-rw-r--r--   1 bin      bin         1155 Jun 28 02:30 ttymon
-r--r--r--   1 root     sys          437 Jun 28 03:36 workstations
```

```
#
```

The */etc/default/cron* file contains the *CRONLOG* variable, which defines whether the execution of regular cron service jobs is to be logged in the */var/cron/log* file (see section "Log files").

```
# cat /etc/default/cron
## @(#)cron.dfl  Revision: 1.3   10/3/90 21:36:36
CRONLOG=YES
#
```

The *DUMPDIR* variable in the */etc/default/dumpsave* file defines the directory to which the contents of main memory are dumped after a system crash (see section "Core dumps").

The */etc/default/inet* file contains settings for variables which control how the system operates in a networked environment. These variables are important if your system is attached to a LAN (local area network) or the NIS (Network Information Service). For further information refer to the manual "Network Administration" "[8]".

```
# cat /etc/default/inet
#ident "$Header: inet,v 1.10 1996/10/24 $"
STATE=active
STANDARDIF=et0:139.22.112.164
EXITIF=
INTERFACES="$STANDARDIF"
OLDBROADCAST=yes
RWHOD=no
DEFAULTGATEWAY=
GATEWAY=no
GATED=no
GLOBALPW=no
TIMESYNC=no
DOMAIN=qm2
YP_MODE=client
AUTO_BINDING=yes
SECURERPC=no
#
```

If the *PANICBOOT* variable in the */etc/default/init* file is set to *YES*, the computer will attempt to reboot the operating system when a panic (unrecoverable error) occurs. By default the system does not reboot after a crash.

```
# cat /etc/default/init
#ident "@(#)$Header: init.dfl,v 1.3 94/07/04 $ SNI"
# @(#)init.dfl  Revision: 1.4   10/3/90 23:06:33
PANICBOOT=YES
#if ulimit shall not be unlimited, uncomment the following line
#ULIMIT=32768
#
```

The *LANG* variable in the */etc/default/language* file sets the default native language for the login environment of your users; the example below selects German. Users can override this default by redefining *LANG* in their own *.profile* files.

```
# cat /etc/default/language
LANG=De_DE.646
#
```

The */etc/default/login* file contains definitions of values governing the environment of a user after login. This file is evaluated before */etc/profile* when the user logs in. As system administrator you must watch out for errors such as duplicated path name settings in the two files, resulting in unnecessarily long path names.

The *TIMEZONE* variable affects the system time. In the example below it is commented out because the value is taken from the */etc/TIMEZONE* file. If you change the value in */etc/default/login*, you must make the same change in */etc/TIMEZONE* , as otherwise inconsistencies will arise; so you should set *TIMEZONE* in */etc/TIMEZONE* only.
The *ULIMIT* variable defines the maximum file size (in 512-byte blocks). In the example below the value is calculated depending on the size of main memory.

```
# cat /etc/default/login
#ident "@(#)$Header: /cvs/CVSROOT/networking/commands/bin/neu/login
v 1.1.1.1 1997/05/02 12:01:27 cvs EXP $ SNI
#CONSOLE=/dev/console
#TIMEZONE=EST5EDT
HZ=100
ULIMIT=` expr \` memsize\`  '/' 512`
PASSREQ=YES
ALTSHELL=YES
PATH=/usr/bin:/usr/sbin:/opt/bin:/usr/ccs/bin
SUPATH=/sbin:/usr/sbin:/usr/bin:/opt/bin:/usr/ccs/bin
...
#
```

Variable	Meaning
ALTSHELL	If set to *YES*, the *SHELL* variable may be assigned values other than the default shell defined in */etc/passwd* .
CONSOLE	If set, the specified terminal is the only one at which the system administrator is allowed to log in as *root*.
HZ	Defines the system clock rate (number of clock ticks per second).
PASSREQ	If set to *YES*, all users must have a password (with the exception of user *root*).
PATH	Path specification for all nonprivileged users.
SUPATH	Path specification for all users with a user ID of 0 (does not apply to the *su* command).
TIMEZONE	Defines the timezone. If a value is defined here, the same value must be used in */etc/TIMEZONE*.
ULIMIT	Defines the maximum file size (in 512-byte blocks).

Table 24: Variables and their meaning in /etc/default/login

In Reliant UNIX Version 5.4x and above it is possible to directly read and write MS-DOS-formatted floppy disks. In the */etc/default/msdos* file you can define how MS-DOS drive names are mapped to Reliant UNIX special file names. For further information on the MS-DOS functionality provided by Reliant UNIX refer to section "Data interchange with DOS".

```
# cat /etc/default/msdos
#ident "@(#)$Header: msdos,v 1.4 95/08/02 13:54:32 $ SNI"
#ident  "@(#)sco:msdos  1.3"
#ident "$Header: msdos,v 1.4 95/08/02 13:54:32 $"
#
# This file controls the mapping of UNIX
# devices to MS-DOS drive letters and types.
...
A=/dev/at/flp/rfx3ht
#
```

The variables in the */etc/default/passwd* file define password attributes (also refer to section "Format and function of the /etc/shadow file").
The *MAXWEEKS* variable defines the maximum number of weeks for which a password can be left unchanged; *MINWEEKS* defines the minimum number of weeks before a password can be changed. *WARNWEEKS* defines how many weeks in advance the user must be warned that a password is about to expire; *PASSLENGTH* governs the minimum number of characters in a password.
This file has a major influence on security and should be modified only after careful consideration.

```
# cat /etc/default/passwd
#ident "@(#)$Header: passwd.dfl,v 1.3 95/09/26 22:32:10 $ SNI"
# @(#)passwd.dfl        Revision: 2.3   91/08/28 22:02:41
MAXWEEKS=24
MINWEEKS=0
WARNWEEKS=1
PASSLENGTH=6
#
```

The variables in the */etc/default/su* file define security-related values which are examined when you use the *su* command to switch to the *root* account (also refer to section "su").
The *SULOG* variable defines the file which logs all attempts to switch to another account using the *su* command.
If the *CONSOLE* variable is defined, any attempt to switch to the *root* account using the *su* command will be logged on the specified terminal.
The *PATH* and *SUPATH* variables define the search paths which will be set after a successful *su* command for a nonprivileged account and for the *root* account respectively.

```
# cat /etc/default/su
#ident "@(#)$Header: su.dfl,v 1.9 96/10/14 11:17:40 $ SNI"
...
SULOG=/var/adm/sulog
#CONSOLE=/dev/console
PATH=/usr/bin:/usr/sbin:/opt/bin:/usr/ccs/bin
SUPATH=/sbin:/usr/sbin:/usr/bin:/opt/bin:/usr/ccs/bin
#
```

Variable	Meaning
SULOG	All attempts to use *su* to switch to another account are logged in the specified file.
CONSOLE	If set, any attempt to use *su* to switch to the *root* account will be logged on the specified

	terminal.
PATH	Path specification which applies after successful execution of the *su* command for a nonprivileged user account.
SUPATH	Path specification which applies after successful execution of the *su* command for the *root* account.

Table 25: Variables and their meaning in /etc/default/su

The */etc/default/tar* file can be used to define default special file names for use with the *tar* command. The number following the word *archive* can be used as an option in the command, and the corresponding device will then be accessed automatically. You can adapt the entries in */etc/default/tar* to match the available floppy disk formats, streamer drives and other devices.
On the basis of the value *archive1=/dev/ios0/rstape00*3 as set in the sample file below, for example, the command *tar -cv1 /home* could be substituted for *tar -cvf /dev/ios0/rstape003/home*.

```
# cat /etc/default/tar
#ident "@(#)$Header: tar.dfl,v 1.1 1992/11/16$ SNI"
#       $Source: /N2A0006/usr/src/cmd/tar/a20r/RCS/tar.dfl,v $
...
#          device               block   size    istape
archive0=/dev/ios0/rstape003h   20      525000  t
archive1=/dev/ios0/rstape003    20      150000  t
archive4=/dev/ios0/rstape003hn  20      525000  t
archive5=/dev/ios0/rstape003n   20      150000  t
archive7=./tarfile              1       0       n
archivef=/dev/null              1       0       n
#
```

The */etc/default/boot* file is used to set defaults for the partition and disk from which the kernel is booted, the partition and disk on which the *root* file system is set up and the partition and disk containing the swap area. The file is created during installation and filled with the corresponding values.
If virtual disks are used, the *bootflags* and *boot* parameters are set via the */etc/default/boot* file. The relevant values for these parameters are added when the virtual disks are configured. The syntax differs between RM200/RM300/RM400 systems (first example) and RM600 systems (second example).

```
# cat /etc/default/boot
bootflags=0x0
boot=ios0/sdisk000s0
root=ios0/sdisk000s0
swap=ios0/sdisk000s1
#
# cat /etc/default/boot
BOOTP0=root=ios0/sdisk000s0
BOOTP1=swap=ios0/sdisk000s1
BOOTP2=bootflags=0x0
BOOTP3=bootdev=ios0/sdisk000s0
#
```

Note:

For further information on the *bootflags* parameter or on virtual disks, refer to the manual "Virtual Disks" "[12]".

5.5.3 The /etc/skel directory

The files in the */etc/skel* directory are copied into the login directory of each new account you create (see the *useradd* command in section "Managing accounts and user groups"). The */etc/skel/.profile* file allows you to provide new users with a predefined working environment.

```
# cat /etc/skel/.profile
# @(#)stdprofile        Revision: 1.3     10/4/90 02:52:52
#       This is the default standard profile provided to a user.
#       Users are expected to edit it to meet their own needs.
MAIL=/usr/mail/${LOGNAME:?}
#
```

You can add other files (*.kshrc* for Korn shell users, for example) or directories to the */etc/skel* directory, and they will all be copied into the login directory of new user accounts. Thus before setting up user accounts on your system you should customize the */etc/skel* directory to save your users extra work.

5.5.4 The /etc/profile file

The */etc/profile* file is a system-wide configuration file. The entries in */etc/profile* are evaluated for all shell users who log in on your system. The evaluation of some entries depends on the status of the user. Before setting certain variables, for example, the system may check whether the user is logging in as *root* or *admin* (i.e. with a user ID of 0).

```
# cat /etc/profile
#ident "@(#)$Header: profile,v 1.32 1997/06/23 $SNI"
# set default file creation mask
trap "" 1 2 3
umask 022
# setup the default timezone
. /etc/TIMEZONE
case "$0" in
-jsh | -ksh | -rsh | -sh)
# be quiet if a .hushlogin exists
if [ ! -z "$HOME" -a ! -f "$HOME/.hushlogin" ]; then
# calculate available disk space in root filesystem.
#echo ""              # skip a line
#. /etc/dfspace
# issue message of the day
trap : 1 2 3
echo ""         # skip a line
if [ -s /etc/motd ]; then
cat /etc/motd
echo ""
fi
trap "" 1 2 3
# check mailbox and news bulletins
if [ "$LOGNAME" != root ] && news -q; then
```

```
echo "new news in /var/news"
fi
if mail -e; then
echo "you have mail"
fi
fi
...
#
```

As system administrator you can make life easier for your users by adapting */etc/profile* , for example by setting paths for new programs or shell scripts.
Make sure, however, that in customizing */etc/profile* you do not actually make things more difficult for individual users. You should also notify your users of any changes you make, typically by using the */etc/issue* or */etc/motd* file or electronic mail (*mail* command). Also refer to section "Communicating with users").

5.5.5 The $HOME/.profile file

When a user logs into an alphanumeric terminal, the global */etc/profile* file is evaluated, followed by the account-specific *$HOME/.profile* file. The *$HOME/.profile* file is created automatically when the account is set up, and it is assigned the defaults defined in the */etc/skel/.profile* file (see section "The /etc/skel directory").

Note:

If a user logs in to a system with a graphical user interface, other files may be interpreted depending on this interface. In this case, consult the documentation of the relevant user interface to find out which files are interpreted.

5.6 Working with file systems

In working with file systems it is essential for you to understand the relationship between hard disks, partitions and file systems. This topic was dealt with in depth in section "File systems".
A file system in Reliant UNIX 5.4x generally corresponds to a partition on a hard disk (although the file system may be smaller than the partition). To access the partition and hence, of course, the file system you use commands which implicitly or explicitly specify the associated special file.

5.6.1 Setting up file systems

New file systems must be set up when the operating system is installed, after extra hard disks have been added, when the hard disk partition layout is completely reorganized, or after major malfunctions. It also possible to set up a file system on a floppy disk to allow data to be transferred to it. In all these cases the commands to use are *mkfs* and *newfs*.
When you install the operating system, the default file systems are set up under menu control. Only in exceptional circumstances should you repartition the hard disk later, as it is a very complex and time-consuming procedure.
The *mkfs* command constructs a new file system on a partition or a floppy disk. The new file system is equipped with a top-level (*root*) directory containing the entries "." and ".." and a *lost+found* subdirectory.

Warning:

When you create a new file system, any data already on the partition or floppy disk is lost.

The following is the recommended procedure for setting up file systems on a hard disk:

- Run the *shutdown* command to switch to single-user mode. This step is not absolutely essential, but should be included to ensure data security.
- Back up any data which is still on the file system (see section "Backup commands").
- Find out the size of the partition, typically using *mkfs -m*.
- If necessary, remove the file system from the directory hierarchy using the *umount* command.
- Set up the new file system using the *mkfs* command.
- Check the new file system using the *fsck* command.
- Add the new file system to the directory hierarchy using the *mount* command.

```
# cd /
# shutdown -y -g30 -i1
# cd /home
# tar -cvf /dev/ios0/rstape003 .
...
#
# mkfs -m /dev/ios0/rsdisk000s4
mkfs -F ufs -o nsect=85, ntrack=9, bsize=16384, fragsize=2048, cgsize
=32, free=10, rps =90, nbpi=2048, opt=t, apc=0, gap=0 /dev/ios0/rsdis
k000s5 683908
# umount /home
# mkfs -F ufs /dev/ios0/rsdisk000s4 683908 85 9
# fsck -F ufs /dev/ios0/rsdisk000s4
```

```
# mount -F ufs /dev/ios0/rsdisk000s4 /home
...
#
```

sysadm interface		
Menu item	**Function**	**Shell command**
file_systems - make	Setting up file systems	mkfs

Table 26: Setting up a file system with sysadm

5.6.2 Mounting file systems

Reliant UNIX uses the Network File System (nfs) to allow access to the data on a computer in the network.
You can make file systems or parts of file systems available to other users to mount them in their file system or to mount file systems of other hosts in your file system.
You can make these file systems available or mount them

- automatically after every restart of your system
- explicitly during a session by entering a command on the command line

By default, nfs is started when you enter run level 2 (multi-user operation). When the system leaves run level 2, all mounted resources are unmounted automatically, access to resources which have been made available is denied and nfs is stopped.

Granting access to file systems

The resources entered in the */etc/dfs/dfstab* file are automatically made available when nfs is started.
Each line in the */etc/dfs/dfstab* file contains a *share* command, i.e. the command with which you explicitly grant access to a resource. The *share* command is described in the following section.

```
# cat /etc/dfs/dfstab
...
share -F nfs -o rw /home/chrissi
share -F nfs -o ro /export/public
...
```

The *share* command allows you to explicitly grant other computers access to a file system. The following example grants access to the files located under the */home/chrissi* directory on host *edoardo*. Option *-o rw* grants read and write privileges.

```
# share -F nfs -o rw /home/chrissi
```

If you call the *share* command without options, you obtain information on the file systems currently made available on your system.
Each file system made available with *share* is registered in the */etc/dfs/sharetab* file.

```
# cat /etc/dfs/sharetab
/home/usr     -     nfs     rw,root=antares     Online
/home1     -     nfs     rw,root=antares     Production
#
```

Note:

The */etc/dfs/sharetab* file only reflects the current status. Editing this file has no effect on the file systems currently available but only falsifies the information about which file systems are shared.

With the *shareall* command you can make several resources available at a time. If you do not specify any file with resources to be mounted with *shareall*, the */etc/dfs/dfstab* file is interpreted. If you specify the resource to be made available in a file, you must make the entries using the same format as the entries in the */etc/dfs/dfstab* .
With the *unshare* command, you can cancel the access privilege for a specific resource. The *unshareall* command is used to deny access to all the resources currently made available.

Mounting file systems
The file systems registered in the */etc/vfstab* file are mounted automatically when the system is started.

```
# cat /etc/vfstab
...
/dev/ios0/sdisk000s3   /dev/ios0/rsdisk000s3   usr   ufs   0   yes
/dev/ios0/sdisk000s4   /dev/ios0/rsdisk000s4   var   ufs   0   yes
...
edoardo:/home/chrissi - /home/antonello nfs - yes rw
oscar:/home/group - /home/antonello/dat1 nfs - yes ro
leo:/export/public1 - /home/antonello/public nfs - yes rw
...
```

The *mount* command is used to explicitly mount file systems in your directory. With the *mountall* command you mount several file systems at a time. If you use the *mountall* command without specifying a file which defines the file systems to be mounted, the */etc/vfstab* file is evaluated. In the following example the files under the file system */home/chrissi* on host *edoardo* are mounted under the mount point */home/antonello*.

```
# mount -F nfs edoardo:/home/chrissi /home/antonello
```

Every file system mounted with the *mount* command is registered in the */etc/mnttab* file. When a file system is unmounted with the *umount* command, the entry is removed.

```
# cat /etc/mnttab
/dev/root       /       ufs     rw,suid 790241000
/proc   /proc   proc    rw      790241000
/dev/fd /dev/fd fdfs    rw      790241000
/dev/ios0/sdisk000s2    /opt    ufs     rw,suid,noquota 790241003
/dev/ios0/sdisk000s3    /usr    ufs     rw,suid,noquota 790241004
/dev/ios0/sdisk000s4    /var    ufs     rw,suid,noquota 790241004
/dev/ios0/sdisk000s5    /home   ufs     rw,suid,noquota 790332702
qm2drive:/home/FDS.V4   /home/FDS.V4    nfs     ro      790332971
#
```

Note:

The */etc/mnttab* file only reflects the current status. Editing this file has no effect on the file systems currently mounted but only falsifies the information about which file systems are mounted.

Unmount individual file systems with the *umount* command. To unmount all currently mounted file systems, use the *umountall* command. In the following example, all files under the */home/antonello* directory which are made available from host *edoardo* are unmounted.

```
# umount edoardo:/home/chrissi
```

Note:

For more information on mounting file systems, refer to the manual "Network Administration" "[8]".

sysadm interface		
Menu item	**Function**	**Shell command**
file_systems - mount	Mounting file systems	mount
file_systems - unmount	Unmounting file systems	umount

Table 27: Mounting a file system with sysadm

5.6.3 Checking and repairing file systems

A file system maintains organizational information which is essential to correct file system operation (see section "File systems and file system types"). Hard disk accesses during normal operation, such as when a file is saved, may cause errors which are not immediately noticed. As a result a situation may arise where the organizational information for the file system no longer reflects the actual situation, and this can lead to major malfunctions. Therefore it is essential for a file system to be regularly checked and where necessary repaired.
You should always check a file system before performing a full file system backup, to ensure that no errors and inconsistencies are backed up along with the data (see section "The ufsdump and ufsrestore commands").
If you suspect that there is something wrong with a file system, you must check it with the *fsck* command. You must always specify the type of file system you want to check (on file system types refer to section "File systems and file system types").

Warning:

The *fsck* command must be applied only to unmounted file systems, as otherwise the file system may be corrupted. So before running *fsck*, unmount the file system with *umount*. The only exception is the *root* file system, which must remain mounted at all times.

```
# cd /
# shutdown -y -g30 -i1
# umount /home
# fsck -F ufs -y /dev/ios0/rsdisk000s5
** /dev/ios0/rsdisk000s5
** Already clean
1489 files, 22529 used, 14407 free (151 frags, 18032 blocks, 0.1% fragmentation)
#
```

fsck checks the actual conditions on the partition against the information in the file system organization files. The command works in five phases, or *passes*, and checks the following

items:

- Are there any blocks which are referenced by more than one inode or which are referenced by an inode and by the free block list?
- Are there any blocks which are referenced by an inode or the free block list and are outside the range of the file system?
- Are there any incorrect link counts?
- Are there any incorrect directory sizes?
- Is the inode format correct?
- Are there any unlisted blocks?
- Are there any files which point to an unallocated inode?
- Are there any inode numbers (inumbers) outside the allocated range?
- Are there any directories which do not have "." and ".." as their first entries?
- Are there more blocks for inodes than there are in the file system?
- Is the format of the free block list correct?
- Is the total of free blocks and/or free inodes correct?

sysadm interface		
Menu item	**Function**	**Shell command**
file_systems - check	Checking file systems	fsck

Table 28: Checking a file system with sysadm

5.6.4 Command overview

The following table summarizes the shell commands used for file system processing and matches them with the equivalent menu items in the *sysadm* interface, which are on a submenu of the *file_systems* selection (also refer to section "The sysadm interface").

sysadm interface		
Menu item	**Function**	**Shell command**
file_systems - make	Constructing file systems	mkfs
file_systems - mount	Mounting file systems	mount
file_systems - unmount	Unmounting file systems	umount
file_systems - check	Checking file systems	fsck
file_systems - list	Listing file systems	mount
file_systems - diskuse	Reporting free disk space	df
file_systems -	Listing files by age	find

fileage		
file_systems - filesize	Listing files by size	du
file_systems - defaults	Managing values in */etc/vfstab*	Any editor
file_systems - identify	Identifying file system types	fstyp

Table 29: File system processing with sysadm (command overview)

6 Backing up, restoring and interchanging data

After a system failure, the system administrator must be capable of restoring the computer to its original state without excessive loss of data. This entails performing regular data backups, the subject of the first part of this chapter.
Another aspect of electronic data processing which is growing ever more significant is data interchange. The prevalence of buzzwords like "client/server computing" and "distributed processing" emphasizes this trend. The second part of this chapter deals in depth with data interchange.

6.1 Backing up and restoring data

Reliant UNIX features a range of data backup commands supporting various forms of data backup. Some of these commands are also suitable for interchanging data with other systems (see section "Data interchange between different UNIX systems").
The *cpio*, *tar* and *dd* commands are perfectly adequate for ad hoc backups of low-volume data. If you need to back up an entire computer together with all its file systems, you can turn to additional commands and services:

- For Reliant UNIX 5.4x, Siemens Nixdorf offers the DSSI backup system, which also backs up the root file system and hard disk partitions (see section "DSSI").
- The *ufsdump* and *ufsrestore* commands allow you to back up and restore complete ufs file systems. They are not suitable for online backups (see section "The ufsdump and ufsrestore commands").
- The *vxdump* and *vxrestore* commands allow you to backup and restore complete VxFS files (see also section "The vxdump and vxrestore commands").
- Commercial products with user-friendly interfaces which greatly simplify data backups allow you to back up both individual files and entire file systems (see section "Backing up with NetWorker").

6.1.1 Preparing for data backups

Before performing data backups there are a number of preparations you should make. You must decide which data to back up and when, and which backup medium to use.
What you need to do is draw up a backup strategy tailored to the needs of your users and their applications. A typical simple backup scheme might involve a full monthly backup of all file systems and a weekly incremental backup of user data. In many cases, though, it is also necessary to carry out daily incremental backups of the most important file systems.
In addition, the timing of data backups is highly significant. To guarantee data consistency, the minimum requirement is that no users or processes should be able to access the data you are backing up while the backup is in progress. That means that backups should always be performed in single-user mode.
Be sure to clearly label all the backup media. You must make it easy to see what you have backed up, when you backed it up, and what tool you used. In this way the different versions you have backed up will be uniquely identifiable.
Backup media storage is a further important consideration. Backup media, like the operating system installation set and the application software, must always be kept in a safe place.

The most suitable location is a lockable fireproof cabinet.

6.1.2 Backup media overview

As a rule you will be backing up data on 1/4 inch cartridge tapes (QIC), because with a capacity of 150 MB up to 2 GB they offer sufficient storage capacity. Other backup media include magneto-optical CD drives, DAT or Video 8 cassette drives and removable disks.

Warning:

There are limits to the number of times you can rewrite cartridge tapes and to their useful life. To avoid unnecessary data loss you should follow the manufacturer's instructions on operation and storage.

In principle, all the backup commands discussed below can likewise be used for tape backups. Unlike floppy disks, cartridge tapes do not need to be formatted. To make optimum use of a tape's storage capacity you can store many archives on one tape.
When you place a cartridge in the drive, the tape is automatically wound back to the start. The special file you use determines whether the tape will be rewound on completion of the backup. If the tape is not rewound, a new archive can then be added to the tape. To position to a particular point on the tape you use the *mt* command.
With all tape processing procedures it is important to keep exact records of what you have backed up and where (also refer to section "script").
Owing to their small capacity, floppy disks are not suitable for full-scale data backups, but only for backing up individual files. In principle, however, all the backup commands discussed here can be used for floppy disk backups.
To format floppy disks you use the command *flformat special_file*. The name of the character special file depends on the floppy disk format and on the hardware. The *format* and *flformat* commands are described in the manual "Commands" "[5]"; special files are discussed in this manual in section "Special files".

6.1.3 Backing up with NetWorker

In addition to the standard Reliant UNIX commands and utilities, a number of commercial backup and archiving products are available, some of them featuring graphical interfaces and client/server architectures for heterogeneous environments designed to make the system administrator's job far simpler.
One such product to emphasize is NetWorker, which backs up data from a wide range of hardware platforms on a central server. The server maintains an index of all the files backed up on it, allowing you to search for and select any files you want to restore. You access NetWorker's routines via a graphical interface.
As NetWorker can be used to operate jukeboxes and robots, it is also suitable for unattended automatic high-volume backups.
A NetWorker Backup/Restore system with restricted functionality as compared with the full version has been integrated into the Reliant UNIX operating system version 5.43 and higher, so that you can familiarize yourself with the NetWorker functionality.
You can order NetWorker from your local Siemens Nixdorf office.

6.1.4 Backup commands

This section examines the data backup commands available on Reliant UNIX systems. The following table summarizes these commands and outlines their characteristics.

Command	**Scope of backup**	**Special files, symbolic links**	**Incremental backups**
tar (combined with *find -newer*)	Directory trees	Yes	Only in combination with *find*
cpio (combined with *find -newer*)	Directory trees	Yes	Only in combination with *find*
dd	Physical partition backup	Yes	No
ufsdump ufsrestore	File systems of type ufs only	Yes	Yes

Table 30: Backup commands under Reliant UNIX 5.4x

6.1.4.1 The cpio command

The *cpio* command can be used both for data interchange and for data backup. The names of the files you want to back up must be passed to *cpio* from standard input. You then redirect the output to a file known as an archive file, which can be on a floppy disk, a tape or the hard disk. The following exemplifies the simplest way of using *cpio*:

```
# cd /home/ms
# ls | cpio -ocv > /dev/ios0/rstape003
...
#
# ls | cpio -ocv > /tmp/ms.cpio
...
#
```

This copies all the files listed by the *ls* command to a *cpio* archive on tape in the first instance and to a disk file in the second instance. You should then view the table of contents of the tape or of the archive file to check that the backup has worked properly. This involves input redirection:

```
# cpio -itv < /dev/ios0/rstape003 | more
...
#
# cpio -itv < /tmp/ms.cpio | more
...
#
```

You can restore a *cpio* backup either in full or in part:

```
# cpio -icv < /dev/ios0/rstape003
...
```

```
# cpio -icv < /tmp/ms.cpio
...
```

Note:

If the *-B* option was used when the backup was made, you must also use it when restoring.

If subdirectories were included in the backup, you must select the *-d* option when restoring it.

```
# cpio -icvd < /dev/ios0/rstape003
...
#
```

The *cpio* command is also well suited to copying entire file trees. In the next example all the files and subdirectories in the */home/ms* directory are recursively copied to the */home2/as* directory on a second hard disk:

```
# cd /home/ms
# find . -print | cpio -pdm /home2/as
...
#
```

In combination with the *find* command, *cpio* can be used for incremental backups. In the next example all the files which have been modified in the last 24 hours are backed up:

```
# find . -mtime -1 -print | cpio -ocv > /dev/ios0/rstape003
```

Note:

If you intend to interchange data with a system running a UNIX version older than V5.40, you must include the *-H odc* option. A header written using *-H odc* under UNIX V5.4 is equivalent to a header written using the *c* option in older versions of UNIX.

Note:

For a full description of the *cpio* command refer to the manual "Commands" "[5]".

6.1.4.2 The tar command

The *tar* command is an uncomplicated data backup utility. As with the *cpio* command, you can specify either a device special file or a file on the hard disk as the archive file for *tar*. However, *tar* produces larger archives than *cpio* and works more slowly.
In the following example, all the files in the current directory and its subdirectories are recursively written first to a *tar* archive on tape and then to a disk file.

```
# cd /home/ms
# tar -cvf /dev/ios0/rstape003 .
...
#
# tar -cvf /tmp/ms.tar .
...
#
```

If defaults have been defined in the */etc/default/tar* file, the special file argument can be replaced with the corresponding number (see section "Files in /etc/default").

```
# tar -cv0 .
```

```
...
#
```

You should then view the table of contents of the tape or of the archive file to check that the backup has worked properly.

```
# tar -tvf /dev/ios0/rstape003
...
# tar -tvf /tmp/ms.tar
...
#
```

You can restore a *tar* backup either in full or in part.

```
# tar -xvf /dev/ios0/rstape003 filename
...
# tar -xvf /tmp/ms.tar
...
#
```

Note:

For a full description of the *tar* command refer to the manual "Commands" "[5]".

6.1.4.3 The dd command

You can use the *dd* command both to copy and convert files and to make physical copies of entire partitions. Thus *dd* is particularly well suited to making an exact image copy of a partition, which is typically useful when you want to copy a partition from one hard disk to another. Provided the partition is not too big, it can also be copied to a cartridge tape.

```
# init 1
# killall
# umountall
# dd if=/dev/ios0/sdisk000s4 of=/dev/ios0/sdisk010s4
...
# dd if=/dev/ios0/sdisk000s4 of=/dev/ios0/stape003
...
#
```

Note:

For a full description of the *dd* command refer to the manual "Commands" "[5]".

6.1.4.4 The vxdump and vxrestore commands

The *vxdump* and *vxrestore* commands allow you to back up and restore data in VxFS file systems. *vxdump* backs up either all the directories and files in a file system or only those which have been modified since a given date. There are various dump levels (0 through 9) which you can use to implement incremental backups. Thus you can back up all the files in the file system which have changed since the last time you made a *vxdump* backup at a lower dump level. (For example: if you make a level 2 backup on Monday and a level 4 backup on Tuesday, a level 3 backup on Wednesday will copy all the files which have been added or modified since Monday's level 2 backup. A level 0 backup backs up the entire file system.)

Note:

As *vxdump* does not process an end-of-tape signal, the tape capacity of the storage medium must be specified.
For backing up and restoring ufs file systems you can use the *ufsdump* and *ufsrestore* commands (see section "The ufsdump and ufsrestore commands").

In the following example the whole of */home* partition is backed up on cartridge tape, with no more than 130 MB being written to one tape:

```
# init 1
# killall
# umount /home
# fsck -F vxfs /dev/ios0/rsdisk000s5
...
# vxdump 0Muf 130 /dev/ios0/rstape003 /home
VXDUMP: (Note) Max calculated capacity for each tape is about 129.5MB
VXDUMP: Date of this level 0 dump: Thu Sep 18 10:59:22 1997
VXDUMP: Date of last level 0 dump: the epoch
VXDUMP: Dumping /dev/ios0/rsdisk000s5 (/home) to /dev/ios0/rstape003
VXDUMP: mapping (Pass I) [regular files]
VXDUMP: mapping (Pass II) [directories]
VXDUMP: estimated 133116 blocks (66558 KB) on 0.526 tape(s).
VXDUMP: dumping (Pass III) [directories]
VXDUMP: dumping (Pass IV) [regular files]
VXDUMP: 48.37% done, finished in 0:05
VXDUMP: 96.16% done, finished in 0:00
VXDUMP: 135584 blocks (67792 KB) on 1 tape
VXDUMP: VXDUMP IS DONE
VXDUMP: level 0 dump on Thu Sep 18 10:59:22 1997
VXDUMP: Closing tape device
#
```

The *vxdump* command options used in the above example have the following meanings:

Option	Meaning
0	Dump level for a full backup (numbers between 1 and 9 can be used for incremental backups)
M	Capacity of the storage medium
u	For each file system which was backed up successfully, an entry is added to the */etc/dumpdates* file. This entry records the name of the file system, the backup level and the date.
f	Name of the dump file, in the example */dev/ios0/rstape00*3, which means that the backup is written to cartridge tape.

Table 31: Options of the vxdump command (selected)

If you use the *-u* option, the following information relating to your backup is recorded in the */etc/dumpdates* file: the name of the partition backed up, the dump level (0=full backup) and the date of the backup:

```
# cat /etc/dumpdates
/dev/ios0/rsdisk000s5      0 Thu Sep 18 10:59:22 1997
#
```

When restoring files and directories you can either restore the entire file system or restore selectively, i.e. choosing individual files or directories and their sub-directories.
In the next example the deleted files */home/arthur/file1* and */home/arthur/file2* are to be restored. This entails using the *-x* (extract) option and specifying the backup medium and the relative path name of the files you want to restore.

```
# cd /home
# vxdump -xf /dev/ios0/rstape003 arthur/file1 arthur/file2
Warning: ./arthur: File exists
You have not read any tapes yet.
Unless you know which volume your file(s) are on you should
start with the last volume and work towards the first.
Specify next volume #: 1
#
```

If you do not know the file and path names of the files, you can restore the files interactively:

```
# vxrestore -if /dev/ios0/rstape003
vxrestore > ?
Available commands are:
ls [arg] - list directory
cd arg - change directory
pwd - print current directory
add [arg] - add 'arg' to list of files to be extracted
delete [arg] - delete 'arg' from list of files to be extracted
extract - extract requested files
setmodes - set modes of requested directories
quit - immediately exit program
what - list dump header information
verbose - toggle verbose flag (useful with ''ls'')
help or '?' - print this list
If no 'arg' is supplied, the current directory is used
vxrestore > ls
.:
arthur/    lost+found/
vxrestore > cd arthur
vxrestore > ls
./arthur:
.profile    file1     file2
vxrestore > add file1
Warning: ./arthur: File exists
vxrestore > add file2
vxrestore > extract
You have not read any tapes yet.
Unless you know which volume your file(s) are on you should start
with the last volume and work towards the first.
Specify next volume #: 1
vxrestore > quit
#
```

Note:

For a full description of the *vxdump* and *vxrestore* commands refer to the "System Administrator's Reference Manual" "[7]".

6.1.4.5 The ufsdump and ufsrestore commands

ufsdump and *ufsrestore* are important commands for backing up and restoring data in ufs file systems. *ufsdump* backs up either all the directories and files in a file system or only those which have been modified since a given date. There are various dump levels (0 through 9) which you can use to implement incremental backups. Thus you can back up all the files in the file system which have changed since the last time you made a *ufsdump* backup at a lower dump level. (For example: if you make a level 2 backup on Monday and a level 4 backup on Tuesday, a level 3 backup on Wednesday will copy all the files which have been added or modified since Monday's level 2 backup. A level 0 backup backs up the entire file system.)

In the following example, relating to a MIPS-based system, the whole of partition 5 (mount point */home*) is backed up on cartridge tape, with no more than 130 MB being written to one tape:

```
# init 1
# killall
# umount /home
# fsck -F ufs /dev/ios0/rsdisk000s5
...
# ufsdump 0uMf 130 /dev/ios0/rstape003 /dev/ios0/rsdisk000s5
UFSDUMP: (Note) Max calculated capacity for each tape is about 129.5MB
UFSDUMP: Date of this level 0 dump: Thu Jan 19 10:59:22 1995
UFSDUMP: Date of last level 0 dump: the epoch
UFSDUMP: Dumping /dev/ios0/rsdisk000s5 (/home) to /dev/ios0/rstape003
UFSDUMP: mapping (Pass I) [regular files]
UFSDUMP: mapping (Pass II) [directories]
UFSDUMP: estimated 95556 blocks (47778 KB) on 0.378 tape(s).
UFSDUMP: dumping (Pass III) [directories]
UFSDUMP: dumping (Pass IV) [regular files]
UFSDUMP: 69.12% done, finished in 0:02
UFSDUMP: 95552 blocks (47776 KB) on 1 tape
UFSDUMP: UFSDUMP IS DONE
UFSDUMP: level 0 dump on Thu Jan 19 10:59:22 1995
UFSDUMP: Closing tape device
#
```

The *ufsdump* command options used in the above example have the following meanings:

Option	Meaning
0	Dump level for a full backup (numbers between 1 and 9 can be used for incremental backups)
u	The date of the backup is recorded in */etc/dumpdates*
M	Capacity of the storage medium
f	Name of the dump file, in the example */dev/ios0/rstape00*3, which means that the backup

	is written to cartridge tape. (The dump file may also be a file on your hard disk, such as */tmp/dump.out* .)

Table 32: Options of the ufsdump command (selected)

If you use the *-u* option, the following information relating to your backup is recorded in the */etc/dumpdates* file: the name of the partition backed up, the dump level (0=full backup) and the date of the backup:

```
# cat /etc/dumpdates
/dev/ios0/rsdisk000s5      0 Thu Jan 19 10:59:22 1995
#
```

When restoring files and directories you can either restore the entire file system or restore selectively, i.e. choosing individual files or directories and their sub-directories.
In the next example the deleted files */home/arthur/file1* and */home/arthur/file2* are to be restored. This entails using the *-x* (extract) option and specifying the backup medium and the relative path name of the files you want to restore.

```
# ls -l /home/arthur
total 26
drwxr-xr-x   3 arthur   si51        512 Apr 20 14:44 bin
drwxr-xr-x   8 arthur   si51        512 Apr 18 15:07 infolib
drwxr-xr-x   8 arthur   si51        512 Apr 18 15:10 lib
drwxr-xr-x   2 arthur   si51        512 Apr 19 08:04 ow3_patches
# cd /home
# ufsrestore xf /dev/ios0/rstape003 arthur/file1 arthur/file2
Warning: ./arthur: File exists
You have not read any tapes yet.
Unless you know which volume your file(s) are on you should
start with the last volume and work towards the first.
Specify next volume #: 1
#
```

If you do not know the file and path names of the files, you can restore the files interactively:

```
# ufsrestore -if /dev/ios0/rstape003
...
ufsrestore > ?
Available commands are:
ls [arg] - list directory
cd arg - change directory
pwd - print current directory
add [arg] - add 'arg' to list of files to be extracted
delete [arg] - delete 'arg' from list of files to be extracted
extract - extract requested files
setmodes - set modes of requested directories
quit - immediately exit program
what - list dump header information
verbose - toggle verbose flag (useful with ''ls'')
help or '?' - print this list
If no 'arg' is supplied, the current directory is used
ufsrestore > ls
.:
arthur/    lost+found/
ufsrestore > cd arthur
```

```
ufsrestore > ls
./arthur:
.profile    file1     file2
ufsrestore > add file1
Warning: ./arthur: File exists
ufsrestore > add file2
ufsrestore > extract
You have not read any tapes yet.
Unless you know which volume your file(s) are on you should start
with the last volume and work towards the first.
Specify next volume #: 1
ufsrestore > quit
...
# ls -l /home/arthur
total 26
drwxr-xr-x   3 arthur   si51        512 Apr 20 14:44 bin
drwxr-xr-x   8 arthur   si51        512 Apr 18 15:07 infolib
drwxr-xr-x   8 arthur   si51        512 Apr 18 15:10 lib
drwxr-xr-x   2 arthur   si51        512 Apr 19 08:04 ow3_patches
-rw-r--r--   1 arthur   si51       2930 Apr  8 18:09 file1
-rw-r--r--   1 arthur   si51       5979 Apr  8 21:25 file2
#
```

If a file system on a partition has been destroyed and you have run *mkfs* to build a new file system (see section "Setting up file systems"), you can restore the contents of the original file system in full. To do this you use the *-r* (recursive) option:

```
# ufsrestore -rf /dev/ios0/rstape003
...
#
```

Note:

For a full description of the *ufsdump* and *ufsrestore* commands refer to the "System Administrator's Reference Manual" "[7]".

6.1.5 DSSI

The mini-root kernel contains a program named DSSI (Data Storage and System Installation) which is designed to handle both system installation and data backup. DSSI backs up the partitions of all the hard disks installed in the computer. As DSSI is started from the mini-operating system, even backing up the root file system (which cannot be unmounted for backing up) presents no problems.

Note:

DSSI is described in full in the Installation Guide for your system "[2]", "[3]".

6.1.6 Sample backup strategy

This section can do no more than provide recommendations, as there are many different approaches to backing up the entire operating system. Each system administrator has to make his or her own decisions about the criteria to apply and the tools to use.

The */ (root)*, */var, /usr* and */opt* file systems need only be backed up when major alterations have been made, such as when the configuration has been changed, new users added or new software installed. In contrast, the */home* file system, which contains the user data, must be backed up considerably more frequently. Depending on your security requirements, you may need to back it up as often as once a day.
In the following example, only the *cpio* and *vxdump* commands described above are used to perform a backup. The entire base system, consisting of the */* (*root*), */usr*, */opt* and */var* file systems, is backed up using *cpio* ; the user file system */home* is backed up using *vxdump*. A single tape can be used for each file system. If more than one tape should be required for a single file system, all you need to do is select the appropriate options of the backup commands.
Make absolutely sure that the system is in single-user mode. Give your users ample advance warning of imminent interruptions to system operation. It is advisable to make your backups at times when system usage is at a minimum.
Next we briefly take you through each the steps involved in making a backup:

- To perform a backup you have to put the system into single-user mode. Before doing so, you must warn your users so that they have time to close down their programs in the normal way.

```
# cd /
# who
...
# shutdown -y -i1 -g60
...
```

- Next you kill all the processes which are accessing the file systems. You can then run the *ps* command to check that all processes have been stopped. If you find any processes that are still running, you must kill them with the *kill* command.

```
# killall
# ps -ef
...
```

- Once there are no more user or software processes running, you must unmount the file systems. To unmount all the mounted file systems you can use the *umountall* command. (You can use the *mount* command to find out which directories are mounted). The */proc* and */dev/fd* file systems must be unmounted separately. Finally you must remount the */usr* file system so that you can use the *find* and *cpio* commands. The */ (root)* file system always remains mounted.

```
# umountall
# mount
...
# umount /proc
# umount /dev/fd
# mount /usr
```

- Next, for each file system in succession, you check it with the *fsck* command, remount it and back it up. In addition, the names of the files you back up are written to log files which can be printed out later for documentation purposes.
  You must note the password which applied at the time of the backup and keep it in a secure place, because if the backup of the */ (root)* file system ever has to be restored, the old password will be applicable again.

```
# fsck /
...
# find . -mount -print | cpio -ocB > /dev/ios0/rstape003 2> /usr/root.cpio.prot
```

```
...
# umount /usr
# fsck /usr
...
# mount /usr
# insert next tape
# find usr -print | cpio -ocB > /dev/ios0/rstape003 2> usr.cpio.prot
...
# fsck /var
...
# mount /var
# insert next tape
# find var -print | cpio -ocB > /dev/ios0/rstape003 2> var.cpio.prot
...
# fsck /opt
...
# mount /opt
# insert next tape
# find opt -print | cpio -ocB > /dev/ios0/rstape003 2> opt.cpio.prot
...
```

- You back up the */home* file system using the *vxdump* command.

```
# fsck /home
...
# vxdump 0uMf 140 /dev/ios0/rstape003 /home
...
#
```

- On completion of the backup you can reboot the operating system.

```
# shutdown -y -i6
...
```

6.2 Data interchange

Data interchange nowadays plays a major role in electronic data processing. The prevalence of buzzwords such as client/server computing and distributed processing underlines its importance. The term "data interchange" as used in this chapter refers primarily to the data from an application, such as text files. This sort of interchange can be made between operating systems and also between different versions of UNIX. The applications themselves cannot be interchanged without considerable effort: you need to obtain the source code, and then recompile and link it on the other platform.
It is advisable to agree with the recipient of the data beforehand on the method of data interchange. In theory, all the data backup commands (see section "Backup commands") can also be used for data interchange. In practice, though, only the *cpio* and *tar* commands are used. The reason for this is that file systems with a fixed directory structure are rarely interchanged. You will mostly be interchanging files in a directory or subtrees of a directory structure, and *cpio* and *tar* are the most suitable tools for the job.
There is one rule you should always observe: always use relative path names, not full (absolute) path names when storing files for data interchange.

Warning:

> If files which were backed up with their absolute pathnames are restored, the corresponding directory is created or, if it already exists, overwritten. If files which were backed up with the relative pathname are restored, the recipient can choose the directory in which the files are to be restored.

The recipient of the cartridge tape written in the next example will doubtlessly be very grateful for being able to extract the files without having to create the directory structure */home/ms/data/may9*7 first.

```
# cd /home/ms/data/may97
# tar -cvf /dev/ios0/rstape003 .
...
#
```

It is also possible to interchange data with computers with a DOS operating system.

6.2.1 Data interchange between different UNIX systems

Data interchange between different UNIX systems is affected among other things by differences in hardware components.
There are, for example, different types of cartridge tape drive which likewise makes data interchange on tape more complicated:

- 1/4 inch cartridge tape drives with 18-track recording and a tape capacity of about 150 MB (QIC150).
- 1/4 cartridge tape drives with 26-track recording and a tape capacity of about 525 MB (QIC-525 format).
- 1/4 inch cartridge tape drives with 30-track recording and a tape capacity of about 1 GB (QIC-1000 format).

Cartridge tape drives can process a specific number of tracks. Drives are upwards- but not downwards-compatible, i.e. it is also possible for a drive to interpret a cartridge which has been written with fewer tracks than the drive itself is capable of processing.
Note, however, that there must be a suitable device driver and hence also a suitable special

file available on the target system, as otherwise data interchange by this route will not be possible in spite of the compatibility between the storage devices.
In view of the potential hardware incompatibilities, data interchange over a network using the *ftp* and *rcp* commands is to be recommended (see section "Data interchange over a network").

6.2.2 Data interchange with DOS

Reliant UNIX 5.4x is shipped with functionality for reading and writing DOS floppy disks included as standard. This makes it possible to exchange data with DOS-based computers on floppy disks. The following table lists the DOS commands included in Reliant UNIX 5.4x:

Command	Function
doscat	Display the contents of a file
doscp	Copy to and from MS-DOS floppy disks
dosdir	List the contents of an MS-DOS floppy disk in DOS format
dosfilt	Make file with language-specific special characters readable under DOS
dosformat	Format an MS-DOS floppy disk under Reliant UNIX
doslabel	Label MS-DOS floppy disk
dosls	List the contents of an MS-DOS floppy disk in Reliant UNIX format
dosmkdir	Create a directory on an MS-DOS floppy disk
dosrm	Delete a file from an MS-DOS floppy disk
dosrmdir	Delete a directory from an MS-DOS floppy disk

Table 33: DOS commands under Reliant UNIX 5.4x

As with all other Reliant UNIX commands which access devices, you have to specify the appropriate special file, e.g. */dev/at/flp/rfx3ht* for a 1.44-MB, 3.5" floppy disk. However, if the */etc/default/msdos* file has been set up accordingly, you can also access a formatted MS-DOS disk by means of the dummy drive name *a:* (see section "Files in /etc/default").

Note:

For a full description of the commands for data interchange with DOS systems refer to the manual "Commands" "[5]".

6.2.3 Data interchange over a network

If your system is part of a networked environment, data interchange over the network is clearly the fastest and most elegant solution. You can transfer the data to the other system using the *rcp* command or the *ftp* service. In the following example, we first use the *ftp*

service to transfer all the files in a directory to a remote system and then use the *rcp* command to copy a *cpio* archive to a remote system. Note that the *rcp* command in this example only works if the login name of the local user has been entered in the *$HOME/.rhosts* file for the target login (e.g. *as*) on the remote system (e.g. *lear*).

```
# ftp othello
Connected to othello.
220 othello FTP server (UNIX(r) System V Release 4.0) ready.
Name (hamlet:root): ebt
331 Password required for ebt.
Password:
230 User ebt logged in.
ftp>dir
150 ASCII data connection for /bin/ls (131.22.122.107,1815) (0 bytes).
total 4
drwxr-xr-x   7 ebt      si511        512 Oct 05 14:37 infothek
226 ASCII Transfer complete.
73 bytes received in 0.01 seconds (7.1 Kbytes/s)
ftp> binary
200 Type set to I.
ftp> prompt
Interactive mode off
ftp> mput *
...
ftp> bye
#
# cd /home/ms/data/may97
# ls | cpio -ocv > /tmp/data.cpio
...
# rcp /tmp/data.cpio as@lear:
#
```

Note:

For a full description of the *ftp* and *rcp* commands refer to the manual "Commands" "[5]".

7 User support duties

One of your principal duties as system administrator is to provide support for system users. In addition to managing the available system resources this involves installing or activating the resources in the first place. These aspects are dealt with in this chapter.

7.1 Managing accounts and user groups

This section begins by discussing the format and function of the files on which user access to the system is based. It then deals with the most important commands used for account and user group management.
Account management is a crucial aspect of system administration, with a significance extending beyond user support requirements and with a major impact on system security. To understand why, simply consider the fact that file and directory access permissions are governed by the group ID number (among other factors). So you must manage accounts and user groups with great care.
As a rule, you should set up a separate account for each user who needs to work on your system. Two users working on the same account will often find themselves getting in each other's way.

Note:

For a full description of account and user group management using the sysadm interface, refer to the manual "System Administration and Hardware Configuration, SYSADM" "[10]". Account and user group management using SINIX/windows is covered in the manuals "SINIX/windows User Environment V3.0 (CDE), Advanced User and System Administrator Guide" "[21]" and "SINIX/windows User Environment V3.0 (SINIX Desktop), Guide for Experts and System Administrators" "[23]".

7.1.1 Format and function of the /etc/passwd file

The */etc/passwd* file is an ASCII file containing a line of information for each account on the system. The fields in each line are separated from each other by a colon. The */etc/passwd* file is basically a simple database containing all the important information governing user access to the system.

```
# ls -l /etc/passwd
-r--r--r--   1 root     other    1242 Jan 12 11:30 /etc/passwd
# cat /etc/passwd
root:x:0:1:0000-Admin(0000):/:
sysadm:x:0:0:System Administration:/usr/admin:/usr/sbin/sysadm
setup:x:0:0:System Setup:/usr/admin:/usr/sbin/setup
daemon:x:1:1:0000-Admin(0000):/:
bin:x:2:2:0000-Admin(0000):/usr/bin:
sys:x:3:3:0000-Admin(0000):/:
adm:x:4:4:0000-Admin(0000):/var/adm:
uucp:x:5:5:0000-uucp(0000):/usr/lib/uucp:
lp:x:7:8:0000-LP(0000):/var/lp:/sbin/sh
```

```
nuucp:x:10:10:0000-uucp(0000):/var/spool/uucppublic:/usr/lib/uucp/uucico
listen:x:37:4:Network Admin:/usr/net/nls:
sync:x:67:1:0000-Admin(0000):/:/usr/bin/sync
install:x:101:1:Initial Login:/home/install:
iocs:x:72:72:I/O-Control System:/opt/iocs:
backup:x:91:91:Targon-Backup:/opt/backup:
formas:x:97:97:FORMAS-User:/opt/formas:
tlog:x:98:98:T-Log:/opt/tlog:
vmsys:x:100:101:FACE executables:/home/vmsys:/sbin/sh
oasys:x:102:1:Object Architecture Files:/home/oasys:/sbin/sh
admin:x:0:2:FACE System Admininstrator:/home/admin:/sbin/sh
nobody:x:103:102:unprivileged user:/nonexistent:/noshell
diagnose:x:0:1:Service-Menu User:/home/diagnose:/opt/bin/diag_sh
sidiag:x:87:1:Unprivileged Field Service User:/home:/sbin/sh
browser:x:106:231:Browser binary building:/home/browser:/usr/bin/ksh
arthur:x:104:1:Arthur Dent:/home/arthur:/sbin/sh
#
```

Each line in */etc/passwd* is divided into seven fields. The contents of these fields are described in the following table:

Field	Contents
1	The first field contains the account name (login name), e.g. *root*.
2	The entry *x* in the second field is a reference to an entry in the */etc/shadow* file which contains the encrypted password for the account.[1]
3	The third field contains the user ID number (uid).
4	The fourth field contains the group ID number (gid).
5	The fifth field is available for comments. Here, for example, you could enter the user's actual name or room and telephone number.
6	The sixth field contains the name of the user's login directory.
7	The seventh field contains the user's startup program. This is usually a shell (such as */sbin/ksh*), but other programs may also be defined as startup programs. If the field is left empty, it defaults to */sbin/sh*; but for security reasons is should not be left empty.
[1] In earlier versions of UNIX, the second field contained the encrypted password. To enhance system security, the password entry is now stored in the /etc/shadow file, which only system programs and users with a uid of 0 (root) can read.	

Table 34: Fields in the /etc/passwd file

You can access the file using shell commands or the sysadm interface in order to add, delete or modify entries (see section "Setting up accounts and user groups", section "Protecting and locking accounts" and section "Modifying and deleting accounts and user groups"). Nonprivileged users have only indirect access to the file, using the *passwd* command to change their passwords.

Warning:

You are strongly advised **not** to work on the file with an editor. If you do find yourself forced to edit the file directly, at least make sure that you make a backup copy before you start and that you run the *pwconv* command when you have finished in order to apply your changes to the */etc/shadow* file as well.

Warning:

The permissions on the */etc/passwd* file must be set to allow all users read access. For security reasons you should under no circumstances grant write access.

7.1.2 Format and function of the /etc/shadow file

The ASCII file */etc/shadow* contains a line corresponding to each account recorded in the */etc/passwd* file. The correspondence is based on the account name (login name). Here, too, each line consists of fields separated from each other by a colon. The most important information in the file is the encrypted password entry for each account.

```
# ls -l /etc/shadow
-r--------   1 root     sys     601 Jan 12 11:30 /etc/shadow
#
# cat /etc/shadow
root:RLgCHNrK6/ip6:9134:0:168:7:::
sysadm:*LK*:6445::::::
setup:*LK*:6445::::::
daemon:NP:6445::::::
bin:NP:6445::::::
sys:NP:6445::::::
adm:*LK*:6445::::::
uucp:NP:6445::::::
lp:*LK*:6445::::::
nuucp:NP:6445::::::
listen:NP:6445::::::
sync::6445::::::
install:*LK*:6445::::::
iocs:*LK*:7581:0:168:7:::
backup:*LK*:7581:0:168:7:::
formas:*LK*:7581:0:168:7:::
tlog:*LK*:7581:0:168:7:::
vmsys:*LK*:::::::
oasys:*LK*:::::::
admin:*LK*:::::::
nobody:*LK*:::::::
diagnose:*LK*:::::::
```

```
sidiag:*LK*:::::::
browser:QaCTNtETbBWPM:9094:0:168:7:::
arthur:c1wmfbsHaHiag:9142:0:168:7:::
#
```

Each line in */etc/shadow* is divided into nine fields. The contents of these fields are described in the following table.

Field	Contents
1	The first field contains the login name, e.g. *root*.
2	The second field contains either the 13-character encrypted password for the account or a lock flag (**LK**) or a flag indicating that the account has not been assigned a password (*NP*).
3	The third field contains the number of days between January 1, 1970, and the last time the password was changed (January 1, 1970, "the epoch", is held to be the day UNIX was "born").
4	The fourth field contains the minimum period of validity for the password in days.
5	The fifth field contains the maximum period of validity for the password in days.
6	The sixth field defines how many days in advance the user must be warned that a password is about to expire.
7	The seventh field contains the number of days an account is allowed to be left unused.
8	The eighth field contains the date when the account becomes invalid and can no longer be used.
9	The ninth and last field is currently not used and is included to allow for future enhancements.

Table 35: Fields in the /etc/shadow file

Warning:

The permissions on the */etc/shadow* file must be set to allow read-only access to *root* alone. For security reasons, i.e. the encrypted passwords, you should under no circumstances grant any other read or write permissions.

7.1.3 Format and function of the /etc/group file

The ASCII file */etc/group* contains a line of information for each user group, consisting of fields separated from each other by a colon. The fields identify the group name, the group password (optional), the group ID number, and the users belonging to the group.
A user who belongs to a number of groups has the option of switching to a different user

group with the aid of the command *newgrp group-name*.

```
# ls -l /etc/group
-r--r--r-- 1 root root 526 Jun 23 11:30 /etc/group
# cat /etc/group
root::0:root
other::1:
bin::2:root,bin,daemon,adm
sys::3:root,bin,sys,adm
adm::4:adm,daemon
mail::6:
tty::7:tty,adm
lp::8:lp
daemon::12:daemon
man::16:
src::17:root
iocs::72:
autoop::90:
backup::91:
formas::97:
tlog::98:
usrother::100:
vm::101:
nobody::102:
si511::231:
guest::103:fb,di,eg,hlg,install,kh,ko,ms,arthur
#
```

Each line in */etc/group* is divided into four fields. The contents of these fields are described in the following table.

Field	Contents
1	The first field contains the group name, such as *root* in the first line above.
2	The second field optionally contains an encrypted group password.
3	The third field contains the group ID number.
4	The fourth field contains a comma-separated list of the users who belong to the group.

Table 36: Fields in the /etc/group file

Warning:

The permissions on the */etc/group* file must be set to allow read access to all users. For security reasons, you should under no circumstances grant any write permissions.

Warning:

You are strongly advised **not** to work on the */etc/group* file directly with an editor. To access the file you should use only the sysadm interface or the *groupadd*, *groupdel* and *groupmod* commands (see section "Setting up accounts and user groups" and section "Modifying and deleting accounts and user groups").

7.1.4 Listing system users

The *listusers* and *logins* commands supply information about users with accounts on the system. The commands take their information from the entries in the */etc/passwd* file (see section "Format and function of the /etc/passwd file").
The *logins* command is typically used for the purpose of identifying security loopholes. In conjunction with the *-p* option it lists all the accounts which are not password-protected. For security reasons you should immediately lock or allocate a password to any such accounts (see section "Modifying and deleting accounts and user groups").

```
# listusers
arthur          Arthur Dent
di
eg
guest
hlg
install         Initial Login
jd
kh
nobody          unprivileged user
oasys           Object Architecture Files
on
vmsys           FACE executables
#
# logins -p
guest           167      other          1      guest account
#
```

Using the sysadm interface you can call up information about system users by selecting *list* from the *users* menu:

sysadm interface		
Menu item	**Function**	**Shell command**
users - list	Listing user accounts	logins

Table 37: Listing user accounts with sysadm

You can also use the *secure* command to check an account for potential security risks. Among other things it reports on accounts with no password protection, accounts with duplicate user ID numbers, and accounts on which no-one has logged in for the last 30 days. For a full description of the *secure* command refer to the manual "Commands" "[5]".

7.1.5 Setting up accounts and user groups

There are two approaches to managing accounts and user groups. You can either set them up and manage them using the *sysadm* interface (see section "The sysadm interface") or use

the shell commands *groupadd* and *useradd*.
When specified without options, the *useradd* command assigns to the new account the default settings defined in the */usr/sadm/defadduser* file. These can be listed with *useradd -D*.

```
# useradd -D
group=other,1 basedir=/home skel=/etc/skel
shell=/sbin/ksh inactive=0 expire=
#
```

If you want a new user to belong to a new group, as in the example below, you first have to define a new group. You do this with the *groupadd* command. The *groupadd* command simply expects you to specify a group ID number and a group name.

Note:

The group ID numbers from 0 through 99 are reserved and must not be allocated using the *groupadd* command.

```
# groupadd -g 500 guest
# useradd -u 301 -g 500 -d /home/guest -s /sbin/ksh -m guest -k /etc/skel
# passwd guest
New password:
Re-enter password:
```

If you want to apply the defaults from the */usr/sadm/defadduser* file, you can leave out the *-d* , *-s* and *-k* options of the *useradd* command. Finally you have to allocate a password, as otherwise the account will remain locked.
Using the sysadm interface you can set up accounts and user groups by selecting *add* from the *user* menu. If you select *defaults* you can modify the defaults in the */usr/sadm/defadduser* file:

sysadm interface		
Menu item	**Function**	**Shell command**
users - add - user	Adding users	useradd
users - add - group	Adding user groups	groupadd
users - defaults	Displaying and modifying defaults for new accounts	Edit*/usr/sadm /defadduser*

Table 38: Adding accounts and user groups with sysadm

7.1.6 Protecting and locking accounts

You can use the *passwd* command to list password attributes, define or delete passwords, or lock an account. The command is most frequently used to allocate a new password to an account.

```
# passwd ms
New Password:
Re-Enter new password:
```

For extra protection you can use the password aging mechanism. This lets you define a

period after which a password becomes invalid and must be changed. In the following example, user *ms* will have to keep the same password for at least 10 days but will have to change it after no more than 50 days.

```
# passwd -x 50 -n 10 ms
```

There are a number of situations where you will need to lock a user's account (if, for example, you discover an account without a password, or you want to back up the data in a single user account, or the user is occupying too much disk space, or you detect that an account is being used for break-in attempts). To lock an account you use the *passwd -l* command.

```
# passwd -l guest
```

You should at all costs avoid simply deleting a password with the command *passwd -d account-name*, as this on its own makes the account accessible without a password check and creates a security loophole.

Using the sysadm interface you can protect and lock accounts by selecting *password* from the *users* menu:

sysadm interface		
Menu item	**Function**	**Shell command**
users - password	Locking and protecting accounts	passwd

Table 39: Locking and protecting accounts with sysadm

7.1.7 Modifying and deleting accounts and user groups

To modify or delete accounts and user groups you can use the *usermod*, *groupmod*, *userdel* and *groupdel* commands. Before starting work you must check that the appropriate conditions are met as indicated below. In addition you must notify any users who are affected by the changes you make.

You will typically need to modify an account if a user would prefer using a different shell or if the account has to be relocated to some other partition owing to disk space bottlenecks.

```
# logins -l guest
guest            404     si511           231
#
# usermod -d /home1/guest -m -s /sbin/ksh guest
#
```

Any account which is no longer being used is a potential security risk and must be deleted. To do this you use the *userdel* command. To be on the safe side, you should back up the data on the account beforehand.

```
# passwd -l as
# cd /home/as
# tar -cvf /dev/ios0/rstape003 .
# userdel -r as
#
```

Using the *groupmod* command you can change the group ID number or the group name. Before doing so, you should notify the users of the accounts affected by the change.

Using the *groupdel* command you can delete a user group. Before deleting a user group you should make sure that there are no accounts allocated to that group.

```
# groupmod -g 500 -n visitor guest
#
# logins -g guest
# groupdel guest
#
```

Using the sysadm interface you can modify and delete accounts and user groups by selecting *modify* or *remove* from the *users* menu.

sysadm interface		
Menu item	**Function**	**Shell command**
users - remove - user	Deleting user accounts	userdel
users - modify - user	Modifying user attributes	usermod
users - remove - group	Deleting user groups	groupdel
users - modify - group	Modifying group attributes	groupmod

Table 40: Modifying and deleting accounts and user groups with sysadm

7.1.8 Defining user disk space quotas

Reliant UNIX 5.4x allows the system administrator to restrict the disk space usage of specific users on ufs file systems by imposing quotas. The quota mechanism operates on the fundamental resources of a file system: its inodes and data blocks. In this way you can prevent file systems overflowing. By imposing a quota on a user you do not alter the value of *ulimit* , which merely places a ceiling on the maximum size of a single file.
The quota may apply to inodes (number of files allowed) and/or to data blocks (disk space occupancy). There is a further distinction between absolute quotas (hard limits) and provisional quotas (soft time limits). In the first case the user cannot go beyond the quota under any circumstances. In the second case the quota can be exceeded temporarily. As many application programs generate very large temporary files such as print files, soft limits are preferable.
When defining quotas for a file system you are advised to adopt the following approach:

- Check that the file system is a ufs file system.
- Unmount the file system using the *umount* command.
- If the file system is listed in the */etc/vfstab* file, change the value of the entry in the *mntopts* (mount options) column to *rq* (also refer to section "The /etc/vfstab file".)
- Remount the file system using the *mount* command.
- Create an empty file named *quotas* in the directory which acts as mount point for the file system.
- Run the *edquota* command for each user individually. This loads the editor defined in the *EDITOR* environment variable, and then you can define the user-specific values (a data block is 512 bytes long).
- Run the *quotacheck* command to check the consistency of the values you have specified

for the file system.

- Enable the quota mechanism by running the *quotaon* command. (You can disable it again with *quotaoff*.)

You should notify all the users involved of the quotas you have imposed. The users themselves should in turn add the *quota* command to their *.profile* files to ensure that they are kept informed of the current status of their quotas.

Note:

For full descriptions of the *edquota*, *quota*, *quotacheck*, *quotaoff* and *quotaon* commands refer to the "System Administrator's Reference Manual" "[7]".

7.1.9 Command overview

The following table summarizes the shell commands used for account and user group management and matches them with the equivalent menu items in the *sysadm* interface, which are on a submenu of the *users* selection (also refer to section "The sysadm interface").

Function	**Shell command**	**sysadm interface**
Setting up user accounts	useradd	users - add - user
Modifying user attributes	usermod	users - modify - user
Deleting user accounts	userdel	users - remove - user
Setting up user groups	groupadd	users - add - group
Modifying group attributes	groupmod	users - modify - group
Deleting user groups	groupdel	users - remove - group
Listing accounts and user groups	logins	users - list
	listusers	
Defaults for new accounts	useradd	users - defaults
Modifying password attributes	passwd	users - password

Table 41: Account and user group management (command overview)

7.2 Task scheduling

Regularly recurring tasks which are designed to run at specific times and require no intervention on the part of the system administrator are performed under the control of *crontab* files. Task execution is then handled by the *cron* daemon, a process which runs permanently in the background. In cases where a task needs to be run only once at a defined time, the *at* command can be used instead. Both the *crontab* file control mechanism and the *at* command are by default available only to the system administrator.
Authorization to use the task scheduling facilities is controlled by four ASCII files, */etc/cron.d/cron.deny* , */etc/cron.d/cron.allow*, */etc/cron.d/at.deny* and */etc/cron.d/at.allow*. If *cron.allow* and/or *at.allow* exist, only users listed in the files are allowed to use the corresponding services. If these files do not exist but *cron.deny* and/or *at.deny* do, all users who are not listed in *cron.deny* or *at.deny* are allowed to use the corresponding services.
As system administrator you should make both services available to your users, but you should also inform them when the system is likely to be off-line. See section "Communicating with users".
In the case of tasks scheduled by means of a *crontab* file, the system must be on-line at the scheduled execution time, as otherwise the task cannot be executed. In contrast, if a task is scheduled using the *at* command and the system is not in service at the scheduled execution time, the task will be performed later when the system is on-line.

7.2.1 The cron service

The *cron* service allows you to have regularly recurring tasks performed at specified times. The service is controlled by the *cron* daemon, which reads its control files, the *crontab* files, once a minute to find out whether there are any tasks for it to perform. Task execution is then logged in the */var/cron/log* file (see also section "Log files").
The *cron* daemon starts up automatically when the system is booted, and you must never invoke it explicitly by command, not even if it fails to start up when you boot the system. In the latter case you must reboot the system (see section "Switching run levels").
The */var/spool/cron/crontabs* directory contains the *crontab* files of the system administrator and other users. You can list the contents of these files with the command *crontab -l user*. There must be only one *crontab* file for each user. *crontab* files can be edited at any time using an editor or the *crontab -e* command.
A script run under the *cron* service should have the environment variables it requires set internally. That means that, for example, the path for all commands subsequently used must be set in the first line of the script, e.g. *PATH=/usr/sbin:/home/murtan/bin:/opt/bin;export PATH*
.

```
# crontab -l
#ident "@(#)$Header: root,v 1.6 95/07/03 10:04:49 cs Exp $ SNI"
17 5 * * 0 /bin/su root -c "/sbin/cleanup > /dev/null"
...
...
# LVS  check size of log files every hour
0 * * * * /opt/SIlvs/bin/lmlogchk
43     *  *  *  *   /usr/sbin/ypxfr_1hour
43     7  *  *  *   /usr/sbin/ypxfr_1day
43  1,12  *  *  *   /usr/sbin/ypxfr_2day
#
```

Each task has a line of its own in the *crontab* file. Each line consists of six fields separated

by blanks. The first five fields must consist of numeric values defining the execution schedule. Each field may contain multiple values, either comma-separated or in the form of a range. An asterisk (*) is a placeholder standing for all legal values. The sixth field contains the command or script which is to be run.
Thus if *0,20,40* were entered in the first field, the script would be run every 20 minutes; and if *1-6* were entered in the fourth field, the script would be run only in the first six months of the year. If there were an asterisk in all the first five fields, the script would be run every minute.
The script or command in the sixth field must be given its full path name. In addition, its standard output and its error channels should be redirected to a file, as otherwise they will be "posted" to the *cron* task submitter by *mail* .

Field	Contents	Legal values
1	Minute(s)	0-59
2	Hour(s)	0-23
3	Day(s) of month	1-31
4	Month of year	1-12
5	Day of week	0-6 (0=Sunday)
6	Command or script	

Table 42: Fields in a crontab file

Using the sysadm interface you can add, modify, delete or list *cron* tasks by selecting items from the *schedule_task* menu:

sysadm interface		
Menu item	**Function**	**Shell command**
schedule_task	Adding, modifying, deleting or listing cron tasks	crontab

Table 43: Managing cron tasks with sysadm

7.2.2 The at command

The *at* command executes commands, or commands from a shell script, at a specified time. A script run by the *at* command sets the current environment variables for the task. The current values for *umask* and *ulimit* and the current directory also apply. Data sent to standard output and to the error channels is automatically "posted" to the command submitter by *mail* .
The arguments expected by *at* are a date and a script name or a command. You can list pending tasks with the *-l* option, and you can delete an *at* task with the *-r* option and the corresponding task number.

```
# at 22:15 < script1
job 780536456.a at Mon Jul 22:15:00 MET 1994
```

```
# at -f /home/murtan/scripts/script2 23:59
job 780536600.a at Mon Jul 23:59:00 MET 1994
# at -l
user = root    780536456.a    Mon Jul 22:15:00 MET 1994
user = root    780536600.a    Mon Jul 23:59:00 MET 1994
# at -r 780536456.a
# at -l
user = root    780536600.a    Mon Jul 23:59:00 MET 1994
#
```

7.3 Package installation

Under Reliant UNIX 5.4x there is a standard procedure for installing software, known as the packaging mechanism. Operating system components and application software alike are shipped and installed as packages.
Using the *pkginfo* command you can display a list of all the packages installed on the system or stored on a data medium. The first column in the list below contains a package category identifier, the second the package name, and the third a brief explanatory label. The example shows the packages which are already installed on a computer.

```
# pkginfo
system      COB85       COBOL Compiler System COB85
application COB85cp     COB85 ColProg support
system      KeyInfo     Siemens Software Authorization Files (Key Infos)
application MAXed       MAXed Full Screen Editor
application ed          Editing Package
...
system      inet        internet utilities
application lp          LP Print Service
utilities   man         Online Manuals
system      motif-r     SNI MOTIF Runtime V 1.1 N7
system      nfs         Network File System Utilities
...
#
```

For detailed information on a package you can use the command *pkginfo -l package-name*. The *package-name* argument refers to the entries in the second column of the *pkginfo* command output.

```
# pkginfo -l face
PKGINST:  face
NAME:  AT&T/SNI Framed Access Command Environment
CATEGORY:  system
ARCH:  R3000
VERSION:  4.3A00
VENDOR:  Siemens Nixdorf Informationssysteme AG
PSTAMP:  karthago940316090642
INSTDATE:  Nov 22 1994 10:20 AM
STATUS:  completely installed
FILES:    951 installed pathnames
73 shared pathnames
268 linked files
131 directories
143 executables
2 setuid/setgid executables
2068 blocks used (approx)
#
```

A package can be installed from one or more floppy disks, from cartridge tape, from CD-ROM or from a network. The installation command is *pkgadd*. In most cases the software is on one or more data media and an authorization floppy (key diskette). The order in which the software and the key diskette are installed depends on the package vendor. If no order is prescribed, you should install the key diskette first.
While installation is in progress, the mechanism checks whether the hard disk capacity is sufficient and whether there are any dependencies with other packages. You may also be prompted to select an installation directory. The complexity of the installation dialog

depends on the package vendor. If you are in doubt over the choice of options, the only solution is to consult the package release notes.
Packages are usually installed immediately; but you can also use the *pkgadd -s* command to initially store the entire software package on your hard disk.

7.3.1 Installing from floppy disks

To install software packages from one or more floppy disks you use the command *pkgadd -d special-file* . Having entered the command you are prompted to insert a disk and type *go* in confirmation. Once the installation script has finished its work, you are informed whether installation was successful.

```
# pkgadd -d /dev/at/flp/f0t
Insert a  diskette into 3.5 inch Floppy0.
Type [go] when ready,
or [q] to quit: go
...
...
#
```

7.3.2 Installing from cartridge tape

To install software packages from cartridge tape you likewise use the command *pkgadd -d special-file* . Having entered the command you are prompted to insert the tape and type *go* in confirmation. Once the installation script has finished its work, you are informed whether installation was successful.

```
# pkgadd -d /dev/ios0/rstape003
Insert a cartdrige tape into Cartdrige Tape Drive.
Type [go] when ready,
or [q] to quit: go
...
...
#
```

7.3.3 Installing from CD-ROM

Software on a CD-ROM containing Siemens Nixdorf add-on products should be installed using the sysadm interface. sysadm accesses the CD-ROM installation routines. To install packages you select the menu item *software_on_CD* .

Note:

For detailed information on installing software from a CD-ROM refer to the manual "System Administration and Hardware Configuration, SYSADM" "[10]".

7.3.4 Software package transfer

A software package can be copied to your hard disk initially and installed later.

```
# pkgadd -s /home/swpkg -d /dev/ios0/rstape003 scde
Insert a cartridge tape into Cartridge Tape Drive.
Type [go] when ready
```

```
or [q] to quit: go
...
Transfer of scde was successful
# pkgadd -d /home/swpkg
...
...
#
```

Note:

You can also install software packages on the hard disk using the *pkgtrans* command.

You can install packages using the sysadm interface by selecting items from the *software* or *software_on_CD* menu as appropriate.

sysadm interface		
Menu item	**Function**	**Shell command**
software - install	Installing packages	pkgadd
software - read_in	Copying packages to hard disk (without installing them)	pkgadd
software - list	Listing packages	pkginfo
software_on_CD - install	Installing packages from CD-ROM	
software_on_CD - pkginfo	Listing the packages on a CD-ROM	

Table 44: Installing and managing packages with sysadm

7.4 Package removal

You can also remove (deinstall) packages from your hard disk. To do this you use the command *pkgrm package-name* . You must make sure that you do not remove a software package which is needed in order for another package the function.

```
# pkgrm scde
...
#
```

You can remove packages using the sysadm interface by selecting *remove* from the *software* menu.

sysadm interface		
Menu item	**Function**	**Shell command**
software - remove	Deinstalling packages	pkgrm

Table 45: Removing packages with sysadm

7.5 System monitoring

To keep your system in full working order, you must monitor the condition of the operating system at regular intervals (at least once per week). Your duties in this respect include the monitoring of log files, disk space and processes.

7.5.1 Disk space

Monitoring the free space on your hard disk is one of your most important duties as system administrator, because an overloaded file system can bring your computer to a halt. The chief commands needed for disk space monitoring are described in section "df and du".

7.5.2 Processes

The term "process" is defined in many different ways in UNIX literature. The simplest definition is that a process is a program in the course of execution. As a program, such as a shell (*sh*), may in turn start (or *spawn*) other processes, on a Reliant UNIX system there are many processes which appear to the user to be running concurrently.
Each process is assigned a unique process identification number (process ID or PID), which is sequentially allocated by the kernel. The kernel is responsible for managing and organizing process execution. Using system calls and signals it controls process creation and running, interprocess communication, and process termination. It also shares out the system resources by allocating execution time slices to each process. As a process is repeatedly executed in memory, but each time only for a fraction of a second, the impression produced is that many processes are being executed simultaneously.
UNIX terminology, often metaphorical in nature, tends to humanize the existence of processes. Processes are spawned and killed, they die or become zombies, and can have parent processes and child processes.
When a process creates (spawns) a new process, the kernel acts as midwife by executing a *fork* system call. The new process is known as a child process, and the calling process is then the parent process. The two processes have different process IDs, but they have the same process environment - environment variables, user ID and group ID - and the same priority, and they respond to the same signals.
The following brief excerpt from a *ps* command report clearly demonstrates the mechanism at work. The *sched* process (PID 0) spawns the processes */sbin/init* (PID 1), *pageout* (PID 2), *fsflush* (PID 3), *kmdaemon* (PID 4), *mr_daemon* (PID 5) and *stidaemon* (PID 6). These processes accordingly all have the same parent process (PPID 0). The */sbin/init* process (PID 1) in turn spawns processes 586 and 1312 and many more besides.

```
# ps -ef
UID   PID    PPID  C    STIME TTY      TIME CMD
root     0       0  0   Mar 13 ?        0:10 sched
root     1       0  0   Mar 13 ?        0:03 /sbin/init
root     2       0  0   Mar 13 ?        0:06 pageout
root     3       0  0   Mar 13 ?        0:06 fsflush
root     4       0  0   Mar 13 ?        0:00 kmdaemon
root     5       0  0   Mar 13 ?        0:00 mr_daemon
root     6       0  0   Mar 13 ?        0:00 stidaemon
root   586       1  0   Mar 13 ?        0:00 /usr/lib/saf/sac -t 300
root  1312       1  0 17:08:40 console  0:01 /usr/lib/saf/ttymon -g -p
Console Login: -d /dev/ttylc -l console
```

```
...
#
```

With the exception of */sbin/init*, processes 0 through 6 are stored in the kernel and will not be found under their names as executable programs on the hard disk. The functions of these processes are described in the following table.

Process	Function
sched	*sched* is the first process started when the operating system is booted. If *pageout* is no longer able to provide sufficient free storage space, *sched* swaps processes in an out. Together with *kmdaemon* and *pageout* it forms the body of the kernel.
/sbin/init	*/sbin/init* starts the initialization of the operating system. Also refer to section "The /etc/inittab file".
pageout	*pageout* is responsible for the majority of the organizational functions of the process management system. Together with *sched* and *kmdaemon* it forms the body of the kernel.
fsflush	*fsflush* is responsible for hard disk I/O operations. It writes the contents of the disk buffer out to hard disk at regular intervals.
kmdaemon	*kmdaemon* is responsible for the majority of the organizational functions of the kernel. Together with *sched* and *pageout* it forms the body of the kernel.
mr_daemon	*mr_daemon* is responsible for monitoring mirrored disks.
stidaemon	*stidaemon* monitors Exabyte devices and moves the tape away from the read/write head if an Exabyte device has not been accessed for a defined period (by default 3 minutes).

Table 46: Important processes and their functions

Using the sysadm interface you can list information about currently running processes or send signals to processes by selecting *processes* from the *performance* menu.

sysadm interface		
Menu item	**Function**	**Shell command**
performance - processes	Listing and sending signals to processes	ps, kill

Table 47: Process management with sysadm

Zombies

Processes are terminated by the *exit()* system call, which is sent to the calling process (parent process). After sending the *exit* system call, a process takes on the zombie state before it is actually terminated by the parent process.
If the parent process ignores the *exit* signal or "hangs", a situation may arise where the child process cannot be terminated normally and continues existing in an "undead" state, like a zombie. This kind of zombie can only be terminated by restarting the system.
A zombie is no longer active as a process and does not actually occupy any system resources, but it is retained in the process table and can be identified with the *ps* command. As processes only remain in the zombie state for a short period of time, *ps* usually finds those zombies which cannot leave the zombie state.
The features that identify a zombie in a *ps* command report are that it has no starting time (no entry in the *STIME* field), that it is not associated with a terminal (no entry in the *TTY* field), and that the entry in the *CMD* field is <*defunct*> .
In earlier versions of UNIX it was possible for a growing number of zombies to cause the process table to overflow. More recent versions are designed such that the kernel automatically releases zombie processes.

```
# ps -ef
UID  PID  PPID C   STIME TTY       TIME COMD
root    0     0 0  Mar 13 ?         0:00 sched
root    1     0 1  Mar 13 ?         5:02 /sbin/init
root    2     0 0  Mar 13 ?         0:01 pageout
root    3     0 0  Mar 13 ?        46:12 fsflush
...
root 1345  1322 0                   0:00 <defunct>
root 1349  1322 0                   0:00 <defunct>
#
```

7.5.3 System failure

Experts like to measure the quality of operating systems by the frequency of system failures. A modern UNIX system should never fail unless there is a serious hardware error. In practice, of course, things are often different. In a complex situation it is possible that the operating system will detect an error and shut itself down immediately (system crash) to prevent consequent data corruption.
In the event of a system failure a message is displayed on the console monitor and the system ceases functioning. The console message will always include the word *PANIC*. If possible, you should note down the exact wording of the message. You should then attempt to reboot the operating system. If that proves impossible, or if the same problem recurs, the cause is generally a hardware error, and you will have to consult customer service.
Software errors can also result in system failures. If the kernel or the operating system is not functioning correctly, there may be a sudden system failure during normal operation. You must treat such events very seriously (especially if they keep occurring), because they may be caused by a bug in the operating system software.
For advice on safeguarding the operating system against system failure refer to chapter "High availability".

7.5.4 Core dumps

After a system crash or a power failure, the entire contents of main memory together with

some additional information are by default stored in the */var/crash* directory (the directory used is defined by the *DUMPDIR* variable in the */etc/default/dumpsave* file, see section "Files in /etc/default"). What happens is that, before the system is rebooted, the contents of main memory are initially dumped to the swap area. From there, in the course of the reboot, the dump is copied to the */var/crash* directory, because if it remained in the swap area it would be overwritten.
You can copy the dump file of the time of the system crash to tape and send it to customer service for analysis.

7.5.5 Log files

Reliant UNIX generates a whole range of log files. As system administrator it is your duty to regularly monitor these files and occasionally truncate some of them to zero length to stop them growing too large. For example:

```
# > /var/adm/log/messages
```

Warning:

You must not delete log files but only set them to zero.

You can "delegate" this duty to the *cleanup* script called by the *cron* service (see section "cleanup" and section "The cron service").

/var/adm/log/messages
The */var/adm/log/messages* file logs system startup messages and error messages generated during normal operation. The contents of this file may provide pointers to possible hardware errors.

```
# more /var/adm/log/messages
...
#
```

/var/adm/sulog
The */var/adm/sulog* file logs all attempts to switch to the *root* account with the *su* command.

/var/adm/lastlog
The */var/adm/lastlog* records the date and time when users last logged in.

```
# who < lastlog
ma       pts/3     Feb 4 07:56
root     console   Feb 4 11:15
ht       pts/4     Feb 4 09:42
#
```

/var/adm/loginlog
The ASCII file */var/adm/loginlog* helps to detect attempted break-ins. Unlike other log files, it exists only if the system administrator explicitly creates it. It must have read and write permissions for its owner *root* only, and it must belong to the user group *sys*. It records login attempts which failed five times in succession. Each line in the file contains the account (login) name, the terminal at which the attempt was made, and the time at which the attempt was made. All system administrators are advised to create this file.

/var/adm/wtmp[x] and /var/adm/utmp[x]

These files maintain logs of all logins and logouts. The *last* command evaluates these files. The files can grow very large and must therefore be regularly truncated to zero length (this can be left to the *cleanup* script, provided that you run *cleanup* regularly). Before truncating the files you can check their contents by viewing them with the *last* | *more* command.

```
# ls -l /var/adm/?tmp*
-rw-r--r-- 1 root root    576 Aug  4 16:16 /var/adm/utmp
-rw-r--r-- 1 root root   5952 Aug  4 16:16 /var/adm/utmpx
-rw-rw-r-- 1 adm  adm    1116 Aug  4 16:16 /var/adm/wtmp
-rw-rw-r-- 1 adm  adm   11532 Aug  4 16:16 /var/adm/wtmpx
#
# ls -l /etc/?tmp*
lrwxrwxrwx 1 root other    13 Jul 13 11:20 /etc/wtmp -> /var/adm/wtmp
lrwxrwxrwx 1 root other    14 Jul 13 11:20 /etc/wtmpx -> /var/adm/wtmpx
#
```

/var/sadm/install/contents

The */var/sadm/install/contents* file is an ASCII file which records all the files and directories installed by the *pkgadd* command. No files installed in any other way are recorded. By applying the *grep* command to this file you can locate an installed file far more quickly than with *find*. You can also check, for example, which files and directories are part of a particular package:

```
# grep SIcdrom /var/sadm/install/contents
/etc/conf/mfsys.d/hs f none 0644 root root 73 4401 785499000 SIcdrom
/etc/conf/pack.d/hs/SP_Driver.o f none 0644 root root 164133 53838
785499000 SIcdrom
/etc/conf/pack.d/hs/space.c f none 0644 root root 259 18193 785499000
SIcdrom
/etc/conf/sfsys.d/hs f none 0644 root root 95 7569 785499000 SIcdrom
...
...
/usr/share/man/mrd/En_US.ASCII/catman/SIcdrom/g7/hs.Z f man_En 0664 bin
man 1696
53164 772508117 SIcdrom
/usr/share/man/mrd/En_US.ASCII/catman/SIcdrom/g8 d man_En 0775 bin man
SIcdrom
/usr/share/man/mrd/En_US.ASCII/catman/SIcdrom/g8/cdrom.Z f man_En 0664
bin man 7
94 22079 772508118 SIcdrom
#
```

/var/cron/log

The */var/cron/log* file is a log of all periodically recurring *cron* tasks (see also section "The cron service"). The file can grow very large and must therefore regularly be truncated to zero length with the command *> /var/cron/log* (this can be left to the *cleanup* script, provided that you run *cleanup* regularly).

/var/crash/*

The */var/crash* directory stores core dumps after a system crash or power failure. The files in it are very large and must be deleted unless they are required for analysis (see also section "Core dumps").

7.5.6 System activity

System utilization can be monitored with the *sar* command (the system activity reporter). The command supplies a wide range of information, with most activities being sampled at defined intervals.
You can also access the *sar* command using the *sysadm* interface by selecting *sar* from the *performance* menu (see section "The sysadm interface").

sysadm interface		
Menu item	**Function**	**Shell command**
performanc e - sar	System activity monitoring	sar

Table 48: System activity monitoring with sysadm

7.5.7 Performance enhancement

If your system comes up against its performance ceiling, typically on account of permanently high utilization, as system administrator you can attempt to enhance system performance. The easiest way to enhance the performance of the operating system is to add appropriate hardware or install better hardware components. You can enhance performance considerably by installing a faster processor or an extra hard disk controller for additional hard disks or by upgrading main memory. Before doing so, though, you must always check what additions and upgrades are supported.
You can also enhance performance by adjusting the tunable system parameters. However, you can generally assume that these parameters have already been set to provide optimum results for most scenarios. To tune these parameters you need a certain degree of expertise. The principal parameters are described in the "Tuning Guide" "[14]".

7.6 Communicating with users

As already repeatedly stressed, communication between system administrator and users is one of your most important duties. Reliant UNIX provides the system administrator with many facilities for communicating with users. This section describes these channels of communication.

7.6.1 Message of the day (/etc/motd file)

The */etc/motd* file is an ASCII file containing a "message of the day" which is displayed on the user's monitor when he or she logs in. The file can be edited with any ASCII editor and is suitable for brief messages. For the message to be displayed, the file permissions must be set to grant all users read access. As the message is a "message of the day", you should change the contents of the file or truncate it to zero length every day. Your users will doubtless be grateful.

```
# cat /etc/motd
Hi, y'all. Today is Wednesday, December 21 -
and here is the phrase of the day:
"All work and no play makes Jack a dull boy"
And so today the computer is shutting down at 5 p.m.
(because we're installing a new hard disk).
#
```

Permanent messages to users should be entrusted to the */etc/issue* file.

7.6.2 Herald screen (/etc/issue file)

The herald screen is what is displayed on the user's monitor before the user logs in. On an alphanumeric terminal the appearance of the herald screen is governed by the ASCII file */etc/issue*.

7.6.3 Broadcast messages (wall command)

The *wall* command broadcasts a message to all users who are currently logged in. It is often used in emergencies, such as when the operating system needs to be shut down at short notice. The message appears on the user's screen. However, as the output of the *wall* command may impair the running of an application or the screen layout of an editor, you should be sparing in your use of this mechanism.

```
# wall
!!!!! ATTENTION PLEASE !!!!!
Owing to a hard disk fault the system will be
shutting down in 5 minutes!!!
Please close all applications immediately!!!
#press [CTRL]-[d]
#
```

7.6.4 Interactive communication (talk command)

The *talk* command allows two users on the same system or on different systems to communicate interactively in an on-screen dialog. In the following example the system administrator wants to talk to user *hlg*, who is on the same system:

```
# talk hlg
```

The user receiving the communication request then sees a message on his or her terminal and signals acceptance of it by typing the following in confirmation:

```
$ talk root
```

The screens of the two parties then divide into two windows, with messages being received in the upper window and sent from the lower window. To terminate the dialog you press the **[DEL]** key.

Note:

For a full description of the *talk* command refer to the manual "Commands" "[5]".

7.6.5 Electronic mail (mail command)

The electronic mail system on installed Reliant UNIX systems is preconfigured to enable users on the system to send each other mail. First, though, they need to know the account (login) names of the other users. As system administrator you can make life easier for your users and yourself by defining aliases and grouping users together.
To customize the mail system you make entries in configuration files, all of which reside in the */etc/mail* directory (see table "Mail system configuration file"). There is no need to modify the files and programs in */usr/lib/mail* and */usr/share/lib/mail* which are also part of the mail system.
In addition to the *mail* program Reliant UNIX 5.4x includes the BSD-based command interface *mailx*. This offers users a more extensive functionality than the ordinary *mail* program.

Note:

For descriptions of the *mail* and *mailx* commands refer to the manual "Commands" "[5]"; for in-depth information on the mail system refer to the "User's Guide" "[6]".

To enable users to enjoy the benefits of e-mail in a networked environment as well, further entries need to be added to the configuration files. These entries relate to the transport system, either *uucp* (unix-to-unix-copy) or *smtp* (simple mail transport protocol), and to information about the network.

Note:

For detailed information on how to network your system refer to the manual "Network Administration" "[8]".

The SINIX/windows Mail Manager provides you with a graphical interface to the functions of the mail system. In addition, the "Communication Manager" product which runs under SINIX/windows combines the functions of the Mail Manager with X.400 services, fax management and MIME support. You can obtain the Communication Manager from your local Siemens Nixdorf office.
The following table summarizes the mail system configuration files:

File	Function
/etc/mail/mailsur	This file defines how user names are

r	interpreted, who is allowed to receive mail, the route by which mail is delivered, and the actions that are to be taken on successful delivery of the mail.
/etc/mail/mailcnfg	This file allows you to set a number of parameters which govern how the *mail* command operates.
/etc/mail/namefiles	This file contains a list of names of files and directories containing alias definitions. (Default contents: */etc/mail/names* and */etc/mail/lists*)
/etc/mail/mailx.rc	This file is a startup file for the *mailx* program.

Table 49: Mail system configuration file

7.6.6 Bulletin board (news command)

The *news* command allows users to read messages and announcements from the system administrator. The advantage of this means of communication is that once a message has been read it will not be displayed again the next time the *news* command is called. Messages are stored as ASCII files in the */var/news* directory. Note the following permission requirements: the directory must have read and execute permission for all users, and the files must be readable by all users. The name of the news file is also the name of the message. Users who do not want to miss out on any news should add the *news* command to their *$HOME/.profile* file (also refer to the example in section "The /etc/profile file").

```
# ls /var/news
Announcement
# news
Announcement (root) Sun Oct  12 19:02:17 1994
From Monday, October 13, you can use the
new PostScript printer.
The command to use is: lp -dpost filename
#
```

7.6.7 Communicating with other systems

Communication with other systems, or with users on other systems, is obviously only possible in a networked environment. The systems need to be equipped with an Ethernet board and connected by cable to a LAN (local area network). In addition, each computer must be allocated a unique Internet number and configured for networked operation. The networking software packages must be installed.
Setting up a local area network is a complex task, and it is usually undertaken by an experienced system administrator who is also a network administrator. On many local area networks there is a special network server (the master NIS server), a slave server and a number of client systems, all linked together by NIS, the Network Information Service. NIS services, formerly known as YP (Yellow Pages) services, ensure that all systems are supplied with the latest network information. That means among other things that system administrators do not all need to modify the network management files on their local

systems whenever a new system is added to the network.

Note:

For detailed information on how to network your system refer to the manual "Network Administration" "[8]".

Various commands are available for communicating in a networked environment. These include *mail* (see section "Electronic mail (mail command)"), and they also include the "r" commands, *rlogin*, *rcp* and *rsh*, and the *telnet* and *ftp* commands, which are available to all users (see also section "Data interchange between different UNIX systems").
Every system administrator of a networked system can also make local file systems or parts of them available to other systems for mounting in their own directory hierarchy. This capability is supported by the *nfs* and *rfs* file system types (see section "File system types: nfs"). In the following example, we start by releasing the file tree below the */home/hamlet* directory on host *othello* for shared access.

```
# share -F nfs -o rw=hamlet:/home/hamlet
```

This file tree is then mounted as a file system on the */home/othello* directory in the directory hierarchy of host *hamlet*. To make the access facility available automatically when the system is rebooted, the */etc/dfs/dfstab* file on *othello* and the */etc/vfstab* file on *hamlet* must be edited accordingly (see also section "The /etc/vfstab file" and section "Mounting file systems").

```
# mount -F nfs othello:/home/hamlet /home/othello
```

7.7 Print services

Under Reliant UNIX you can use both Xprint, the spooler of Siemens Nixdorf, and the AT&T spooler. Xprint must be installed separately. Both spoolers are fully documented in the Reliant UNIX literature. Consequently they are not discussed further in this manual.

Note:

For detailed information on Xprint V5.0, refer to the manuals "Xprint V5.0, User and Administrator Guide" "[15]" and "Xprint V5.0, Reference Manual" "[16]"; the AT&T spooler is described in the manual "SINIX, The AT&T Spooler, LP Print Service" "[17]".

8 System accounting

The accounting utilities on Reliant UNIX systems are a set of C programs and shell scripts which collect system usage data and classify it according to CPU load and utilization by users and processes. They also provide information relating to connect sessions and disk usage.
This chapter describes how the accounting services work. It discusses the main utilities and files that make up the accounting routine. It also includes examples of the reports that the utilities generate.

8.1 Computer Center Accounting Procedure (RAV) for Reliant UNIX systems

In addition to the standard accounting procedures presented in this chapter, Siemens Nixdorf offers a product named RAV-SINIX, which you can order separately. This product provides a user-friendly menu-driven interface to system accounting in a client/server environment.
The RAV Computer Center Accounting Procedure for Reliant UNIX systems allows you to evaluate the data recorded by the standard accounting procedures together with user-specific and accounting-related data recorded in an INFORMIX® database.
Prices defined for each computer service are applied to the accounting data and to the additional information in the database to calculate totals which are billed to users per accounting period.
RAV generates reports and summary lists which provide an overview of the service provider and the service user and of the accrued costs.
You can order RAV from your local Siemens Nixdorf office.

8.2 Overview of the accounting routine

This section summarizes how the accounting system works. The accounting mechanism can essentially be viewed as having four elements:

- While the computer is in operation, data relating to system usage is collected in accounting files.
- Once a day, the *cron* daemon calls the *runacct* program, which processes the various accounting files and produces both daily accounting reports and periodic summaries.
- *runacct* calls the *prdaily* program, which gathers the reports for the day in a file named */var/adm/acct/sum/rprtMMDD*.
- Once per accounting period (typically a month) you can process and output the periodic summaries generated by the *runacct* program by running the *monacct* program (typically with the aid of *cron*; see also section "Accounting routine setup"). The reports generated by *monacct* are stored in the */var/adm/fiscal* directory.

8.3 Accounting methods

There are a number of different accounting methods you can use to generate reports and summaries from the data collected each day: connect accounting, process accounting, disk accounting, fee accounting, and command summary.

8.3.1 Connect accounting

Connect accounting supplies information about all user logins and logouts, about terminal line usage, about the frequency of accounting software activation/deactivation, and about the number of times the system has been rebooted. This entails keeping records of date changes, loading times, accounting software activation and deactivation, run level changes, and the spawning and termination of user, login and *init* processes.

8.3.2 Process accounting

Process accounting supplies the following information with regard to every process running on your system: user and group ID numbers of the process users, process starting time and total runtime, CPU time utilized by the process (this time is divided between the users and the system), occupied disk space, and the terminal controlling the process.

8.3.3 Disk accounting

Disk accounting supplies the following information concerning the files each user has stored on disk: user's login name and user ID number, and the number of blocks occupied by that user's files.

8.3.4 Fee accounting

If you want to bill your users for special services such as file restores, you can use the *chargefee* program. The program syntax is described on the *acctsh*(1M) manual page. The fees are defined in the */var/adm/fee* file. Each entry in this file consists of the user's login name and user ID number together with the appropriate (integer) fee. To bill for special services you call the program with the following arguments: *chargefee login_name fee*

8.3.5 Command summary

A command summary is an overview of resource usage listed by command. You can use this summary to find out which commands have been used most and how these commands utilize your resources.

8.4 Accounting routine setup

To set up the accounting routine you must take the actions described below.

8.4.1 Activating the accounting routine

To have the accounting routine activated when the system goes into multiuser mode, you make a symbolic link between */etc/init.d/acct* and */etc/rc2.d/S22acct*:

```
# ln -s /etc/init.d/acct /etc/rc2.d/S22acct
```

If you do not want accounting data to be collected while the system is being shut down, you make a symbolic link between */etc/init.d/acct* and */etc/rc0.d/K22acct* :

```
# ln -s /etc/init.d/acct /etc/rc2.d/K22acct
```

8.4.2 Customizing system files

In the */var/spool/cron/crontabs/adm* file you must specify the times at which you want the *ckpacct*, *runacct* and *monacct* programs to be executed. The following example illustrates a number of entries. This file should have *root* as its owner, *sys* as its group and *644* as its permissions (for further information on the cron service refer to section "Task scheduling" or to the description of the *crontab* command in the manual "Commands" "[5]").

```
# cat /var/spool/cron/crontabs/adm
#----------------------------entries for adm crontab------------------
#Min Hour Day  Month Day  Command
#         of         of
#         Month      Week
#-----------------------------------------------------------------------
0   *    *    *    * /usr/lib/acct/ckpacct
30  2    *    *    * /usr/lib/acct/runacct 2> /var/adm/acct/nite/fd2log
30  3    1    *    * /usr/lib/acct/monacct
#-----------------------------------------------------------------------
```

In our example, the programs will be executed at the following times:

- *ckpacct*: every hour on the hour
- *runacct*: every day at 2.30 a.m.
- *monacct:* on the first day of every month at 3.30 a.m.

In the */var/spool/cron/crontabs/root* file you must specify the time at which you want the *dodisk* program to be executed.

```
# cat /var/spool/cron/crontabs/root
#----------------------entry for root crontab----------------------
#Min  Hour  Day    Month  Day   Command
#           of            of
#           Month         Week
#------------------------------------------------------------------
30   22    *      *      4     /usr/lib/acct/dodisk
#------------------------------------------------------------------
```

In our example, *dodisk* will be executed every Thursday at 10.30 p.m.

8.4.3 Special discounts for non-prime times

The accounting routine lets you grant discounts to users working on the system at non-prime times. To make use of this feature you first need to update the */etc/acct/holidays* file to reflect the current year and the starting times for prime and non-prime times, plus

public holidays which fall on a working day and any other free working days such as company holidays.

Note:

You must at least update the year parameter, as otherwise the accounting routine will abort and issue a warning. Also remember to update this file at the beginning of each year.

The following example illustrates typical entries in the */etc/acct/holidays* file. Lines beginning with an asterisk are comment lines.

```
# cat /etc/acct/holidays
* Prime/Nonprime Table for Reliant UNIX Accounting System
*
* Curr  Prime  Non-Prime
* Year  Start  Start
*
1994  0700   1900
*
* only the first column (month/day) is significiant.
*
* month/day    Holiday
*
*1/1            New Year's Day (Saturday)
4/1            Good Friday
4/4            Easter Monday
*5/1            Labor Day (Sunday)
5/12           Ascension Day
5/23           Whit Monday
6/2            Corpus Christi
11/1           All Saints' Day
*12/24          Christmas Eve (Saturday)
*12/25          Christmas Day (Sunday)
12/26          Boxing Day
*12/31          New Year's Eve (Saturday)
```

8.5 Accounting routine startup

If you have linked */etc/init.d/acct* to */etc/rc2.d/S22acct*, the accounting routine will be started up automatically every time the operating system is rebooted, specifically when it reaches run level 2 (multiuser mode).
If you have just set up the accounting routine for the first time as described above, there is no need for you to reboot the operating system in order to turn on accounting. All you need to do is enter the following command:

```
# sh /etc/rc2.d/S22acct
```

Note:

If you invoke the accounting routine manually and then get a message indicating that the */var* file system is full, you will have to reboot the operating system after all.

8.6 Accounting reports

You use the *prdaily* and *monacct* programs to produce daily reports and periodic reports respectively.
The daily reports are stored in files named on the pattern */var/adm/acct/sum/rprtMMDD*, where *MMDD* stands for the month and day of the current year. They comprise

- a report on daily line usage
- a report on daily resource usage
- a report on daily command usage
- a report on command usage since the last time the *monacct* program was run
- a report indicating the last login date of each user

The periodic reports are stored in files named on the pattern */var/adm/acct/fiscal/fiscrptMM* and comprise

- a report on resource usage for accounting period *MM*
- a report on command usage for accounting period *MM*
- a report indicating the last login date of each user

The following subsections examine examples illustrating the reports generated by the *prdaily* program. The *monacct* reports have the same format as the *prdaily* reports.
You can print out the files referred to above or process them with your own programs.

8.6.1 Daily line usage report

This report provides information about every terminal line used. The following is a typical example:

```
Oct 18 04:08 1994  DAILY REPORT FOR uranium Page 1
from Mon Oct 17 04:05:03 1994
to   Tue Oct 18 04:05:03 1994
1       runacct
1       acctcon
TOTAL DURATION IS 1440 MINUTES
LINE         MINUTES  PERCENT  # SESS  # ON  # OFF
term/0       1051     73       2       2     2
term/1       262      18       11      6     6
term/2       210      15       30      22    22
TOTALS       1523     --       43      30    30
```

The *from/to* lines indicate the period covered by the report. Then comes a log of the data recorded by the *acctwtmp* program. The second part of the report is a breakdown of the recorded line usage.

Column heading	Meaning
TOTAL DURATION	Total system runtime in multiuser mode
LINE	Terminal line or port by which the system is being accessed
MINUTES	Total line usage time in the accounting period
PERCENT	The *MINUTES* value as a percentage of

	the *TOTAL DURATION* value
# SESS	Number of login accesses to the associated port
# ON	Number of times a user logged in at the associated port
# OFF	Number of times a user logged out at the associated port plus the number of timeouts. Timeouts typically occur at a port the first time the *ttymon* program is called when the system switches to multiuser mode. This column is significant only when the value of *OFF* greatly exceeds the value of *ON*, which is generally an indication that there is something wrong with the terminal connection.

Table 50: Line usage accounting parameters

8.6.2 Daily resource usage report

This report lists system resource usage on a per-user basis. The following is a typical example:

```
Oct 18 04:08 1994  DAILY USAGE REPORT FOR uranium Page 1
LOGIN      CPU (MINS)     KCORE-MINS    CONNECT (MINS) DISK   ...
UID   NAME     PRIME  NPRIME  PRIME   NPRIME  PRIME   NPRIME BLOCKS ...
0     TOTAL    314    108     653     176     31749   16106  1258780...
0     root     22     49      84      153     837     545    519888 ...
2     bin      0      0       0       0       0       0      398776 ...
4     adm      2      2       0       2       0       0      38044  ...
5     uucp     0      0       0       0       27      0      4768   ...
...
```

The meanings of the various columns are:

Column heading	**Meaning**
UID	User ID number.
LOGIN NAME	User's login (account) name.
CPU (MINS)	Time in minutes for which the user process occupied the CPU. This column is subdivided into *PRIME* and *NPRIME* (prime time and non-prime time components). The criteria on which the accounting routine bases this subdivision are defined in the */etc/acct/holidays* file.

KCORE-MIN S	Amount of memory a process requires to run. The value shown refers to the memory segments used in kilobytes per minute. This column is likewise subdivided into prime and non-prime time.
CONNECT (MINS)	Time in minutes for which a user was logged in. If the value in this column is high and the value in the *# OF PROCS* column is low, the user was logged in for a long time but only seldom used the terminal. This column is likewise subdivided into prime and non-prime time.
DISK BLOCKS	This column contains the result of the disk accounting procedure. A block consists of 512 bytes (for accounting purposes).
# OF PROCS	Number of processes invoked by the user. You should look out for high values in this column, as they may indicate a runaway shell script.
# OF SESS	(not shown) Indicates how many times a user logged in.
# DISK SAMPLES	(not shown) Indicates how many disk usage samples were taken to calculate the average disk block rating (i.e. the *DISK BLOCKS* value).
FEE	(not shown) Fees charged to the user.

Table 51: Resource usage accounting parameters

8.6.3 Command summary

Both the daily command summary and the summary of command usage since the last time *monacct* was called classify system resource usage by command. This report shows you which commands are used most and how these commands use the system resources (the data is thus useful as a basis for system tuning). The following is a typical example:

```
Oct 18 04:08 1994  DAILY COMMAND SUMMARY Page 1
TOTAL COAND SUMMARY
COMMAND    NUMBER     TOTAL       TOTAL       TOTAL    MEAN     MEAN    ...
NAME         CMDS   KCOREMIN     CPU-MIN   REAL-MIN  SIZE-K  CPU-MIN ...
TOTALS     238030     506.39      422.05  384381.06    1.20     0.00  ...
vi            487      24.48       12.28    4179.50    1.99     0.03  ...
sh          13508       7.15       12.38   37132.80    0.58     0.00  ...
ftp           144       8.93        4.36     428.26    2.05     0.03  ...
fmli            6       8.61        1.32      45.15    6.53     0.22  ...
cc           1034       7.04        2.21      86.12    3.18     0.00  ...
```

. . .
The meanings of the various columns are:

Column heading	Meaning
COMMAND NAME	Command name. (As the process accounting system deals only with object modules, all shell scripts are covered by the label *sh* .)
NUMBER CMDS	Total number of times the command was called.
TOTAL KCOREMIN	Total amount of memory segments in kilobytes utilized by the command per minute of runtime.
TOTAL CPU-MIN	Total CPU time consumed by the command in minutes.
TOTAL REAL-MIN	Total processing time required for the command in minutes.
MEAN SIZE-K	Mean amount of memory segments in kilobytes utilized by the command per minute of runtime. The value is calculated by dividing *TOTAL KCOREMIN* by *TOTAL CPU-MIN*.
MEAN CPU-MIN	Mean CPU time consumed by the command in minutes. The value is calculated by dividing *TOTAL CPU-MIN* by *NUMBER CMDS* .
HOG FACTOR	Ratio between CPU time and total processing time for a command. The value is calculated by dividing *TOTAL CPU-MIN* by *TOTAL REAL-MIN*.
CHARS TRNSFD	(not shown) Total number of characters transferred by *read* and *write* system calls (contains negative values if it overflows).
BLOCKS READ	(not shown) Total number of physical block reads and writes performed by the command.

Table 52: Command summary parameters

8.6.4 Last login report

This report contains the date a particular login (account) was last used. This data can help you discover unused logins and login directories which you can then archive or delete. The following is a typical example.

```
Oct 18 04:08 1994  LAST LOGIN Page 1
```

```
00-00-00  admin         94-10-15  hjk           94-10-17  anne
00-00-00  autoop        94-10-16  hgf           94-10-17  gs
00-00-00  backup        94-10-16  hoffma        94-10-17  maria
00-00-00  bin           94-10-16  hofu          94-10-17  mecora
00-00-00  daemon        94-10-16  krenz         94-10-17  walter
...
```

8.7 Accounting programs

8.7.1 Accounting routine control programs

startup
The *startup* program (also refer to *acctsh*(1M) in the "System Administrator's Reference Manual" "[7]") is invoked automatically when the system switches to multiuser mode (*init 2*) if a link to *S22acct* has been set up (see section "Accounting routine setup"). It starts up the accounting routine by invoking the *turnacct* program with the *on* option.

shutacct
The *shutacct* program (also refer to *acctsh*(1M) in the "System Administrator's Reference Manual" "[7]") is invoked automatically when the system is shut down, and it terminates the accounting routine by invoking the *turnacct* program with the *off* option.

ckpacct
The *ckpacct* program (also refer to *acctsh*(1M) in the "System Administrator's Reference Manual" "[7]" checks the size of the */var/adm/pacct* file. It is invoked by the *cron* daemon at the time specified in your */var/spool/cron/crontabs/adm* file (see section "Customizing system files"):
- If the */var/adm/pacct* file grows larger than a defined size, *ckpacct* invokes the *turnacct* program with the *switch* option.
- If the number of free disk blocks in the */var* directory drops below 2000, *ckpacct* invokes the *turnacct* program with the *off* option.
- Once there are again 2000 or more disk blocks available in the */var* directory, *ckpacct* invokes the *turnacct* with the *on* option.

turnacct
The *turnacct* program (also refer to *acctsh*(1M) in the "System Administrator's Reference Manual" "[7]") is invoked by *startup* , *shutacct* and *ckpacct*, and also by *runacct* in the SETUP state:
- *turnacct on* turns the accounting routine on by invoking *accton* with the argument */var/adm/pacct*.
- *turnacct off* turns the accounting routine off by invoking *accton* without any arguments.
- *turnacct switch* renames */var/adm/pacct* to */var/adm/pacct?*, where *?* is a sequentially incrementing number, and creates a new */var/adm/pacct* file.

accton
The *accton* program (also refer to *acct*(1M) in the "System Administrator's Reference Manual" "[7]") is invoked by *turnacct* , and it turns process accounting on if called with a file name argument. The accumulating process data is recorded in the specified file. If called without an argument, it turns process accounting off.

nulladm
The *nulladm* program creates an empty file with its permissions set to *664* and its owner and group set to *adm* .

8.7.2 Accounting routine recording programs

acctwtmp

The *acctwtmp* program (also refer to *acct*(1M) in the "System Administrator's Reference Manual" "[7]") is invoked by *startup* and *shutacct*. For connect accounting purposes it keeps records in the */var/adm/wtmp* file for every startup and shutdown of the accounting routine and hence for every system reboot and shutdown.

closewtmp
The *closewtmp* program (also refer to *acct*(1M) in the "System Administrator's Reference Manual" "[7]") is invoked by *runacct* in the SETUP state. For connect accounting purposes it keeps records in the */var/adm/wtmp* file for every user who is currently logged in. Later */var/adm/wtmp* is renamed */var/adm/acct/nite/wtmp.MMDD* by *runacct* and a new */var/adm/wtmp* file is created.

utmp2wtmp
The *utmp2wtmp* program (also refer to *acct*(1M) in the "System Administrator's Reference Manual" "[7]") is invoked by *runacct* in the SETUP state. For connect accounting purposes it creates an entry in the new */var/adm/wtmp* file created by *runacct* for each user who is currently logged in.

lastlogin
The *lastlogin* program (also refer to *acctsh*(1M) in the "System Administrator's Reference Manual" "[7]") is invoked by *runacct* in the CMS state. It updates the */var/adm/acct/sum/loginlog* file, which records the date on which each user logged in. This file is evaluated by *prdaily*.

login, init, ttymon, date
For connect accounting purposes these programs create entries in the */var/adm/wtmp* file for every user login and logout, for every change of run level, for terminal line usage, and for date changes.

exit
For process accounting purposes the kernel records data relating to each running process. Each time a user process ends, *exit* writes this data to the */var/adm/utmp* file.

dodisk
The *dodisk* program (also refer to *acctsh*(1M) in the "System Administrator's Reference Manual" "[7]") is invoked by the *cron* daemon at the time specified in your */var/spool/cron/crontabs/root* file (see section "Customizing system files"). For disk accounting purposes it keeps per-user records of disk accesses and occupied blocks in the */var/adm/dtmp* file. If *dodisk* is invoked with the *-o* option, this data is recorded by the *acctdusg* program. The collected data is consolidated by the *acctdisk* program.

acctdusg
The *acctdusg* program (also refer to *acct*(1M) in the "System Administrator's Reference Manual" "[7]") is invoked by *dodisk -o* . It uses a different *dodisk* algorithm for data capture and activates a slower version of the disk accounting procedure. The data is written to */var/adm/dtmp*.

acctdisk
The *acctdisk* program (also refer to *acct*(1M) in the "System Administrator's Reference Manual" "[7]") is invoked by the *dodisk* program. It merges the data collected by *dodisk* to

form total accounting records in the */var/adm/acct/nite/disktacct* file.

Warning:

The */var/adm/acct/nite/disktacct* file is overwritten each time *dodisk* is invoked.

chargefee

The *chargefee* program (also refer to *acctsh*(1M) in the "System Administrator's Reference Manual" "[7]") is invoked by the system administrator to charge users fees for special services. The fees are defined in the */var/adm/fee* file.

8.7.3 Accounting routine evaluation and display programs

runacct

The *runacct* program is invoked by the *cron* daemon at the time specified in your */var/spool/cron/crontabs/adm* file (see section "Customizing system files"). It evaluates connect, process, disk and fee accounting files and prepares daily summaries and periodic summaries for evaluation by *prdaily* and *monacct*.
When *runacct* is invoked, the */var/adm/acct/nite/lock* and */var/adm/acct/nite/lock1* files are created and */var/adm/acct/nite/lastdate* is checked. The first two files serve to prevent *runacct* being started when it is already running. The *lastdate* file contains the date when *runacct* was last invoked and prevents *runacct* being invoked more than once per day.
While *runacct* is running, */var/adm/acct/nite/active* is created to record its progress and store warnings and error messages. If *runacct* aborts as the result of an error, this file is renamed */var/adm/acct/nite/activeMMDD* and then a new version of */var/adm/acct/nite/active* is created.
While *runacct* is running, all diagnostics are recorded in */var/adm/acct/nite/fd2log*.
runacct runs in different stages known as states. If *runacct* aborts, you can choose the state in which you want it to restart. While *runacct* is running, it writes the name of its current state to the */var/adm/acct/nite/statefile* file. If an error is detected while the program is in a particular state, a message is displayed on the console and sent by mail to *root* and *adm* (see section "runacct error messages").

SETUP
In this state, connect and process accounting files are converted to working files. This involves the following actions:
- The */var/adm/pacct* process accounting file is renamed */var/adm/pacct?* and a new one is created by *turnacct switch* (the *?* stands for an incrementing number appended by *turnacct switch*).
- The */var/adm/pacct?* files created by *turnacct switch* in the current accounting routine are renamed */var/adm/Spacct?.MMDD*.
- The */var/adm/wtmp* connect accounting file is updated and closed by *closewtmp*.
- The */var/adm/wtmp* file is renamed */var/adm/acct/nite/wtmp.MMDD* and a new one is created. The new file is updated by *utmp2wtmp*.

WTMPFIX
- In this state, the */var/adm/acct/nite/wtmp.MMDD* work file is verified for integrity, corrected if necessary and copied to */var/adm/acct/nite/tmpwtmp*.
- Errors are documented in */var/adm/acct/nite/wtmperror*.
- If the program aborts, */var/adm/acct/nite/wtmperror* is renamed

/var/adm/acct/nite/wtmperrorMMDD.

CONNECT
In this state, connect accounting data is merged to form total accounting records. This involves the following actions:
- The data from the corrected working file */var/adm/acct/nite/tmpwtmp* is merged by *acctcon* to form total accounting records in */var/adm/acct/nite/ctacct.MMDD*.
- *acctcon* merges the data in the */var/adm/acct/nite/ctmp*, */var/adm/acct/nite/lineuse* and */var/adm/acct/nite/reboots* files (see section "/var/adm/acct/nite/ctmp").
- Errors are documented in */var/adm/acct/nite/log*.

PROCESS
In this state, *acctprc* merges the process accounting data from the */var/adm/Spacct?.MMDD* files to form total accounting records in */var/adm/acct/nite/ptacct?.MMDD* (see section "acctprc").

MERGE
In this state, *acctmerg* merges the total accounting files generated in the CONNECT and PROCESS states into the daily accounting file */var/adm/acct/nite/daytacct*. This file is evaluated by *prdaily* (see section "acctmerg").

FEES
In this state, *acctmerg* adds the fee accounting data from the */var/adm/fee* file to the daily accounting file */var/adm/acct/nite/daytacct* (see section "acctmerg").

DISK
In this state, *acctmerg* adds the disk accounting data from the */var/adm/acct/nite/disktacct* file to the daily accounting file */var/adm/acct/nite/daytacct* (see section "acctmerg").

MERGETACCT
In this state, *acctmerg* writes the data from the daily accounting file */var/adm/acct/nite/daytacct* to the summary file */var/adm/acct/sum/tacct*, which collects data for periodic summaries. In addition, the daily accounting file */var/adm/acct/nite/daytacct* is copied to */var/adm/acct/sum/tacct.MMDD* so that it can be regenerated if it is lost (see section "acctmerg").

CMS
In this state, *acctcms* merges the command summary data from the */var/adm/Spacct?.MMDD* files (see section "/var/adm/Spacct?.MMDD"). In addition, *lastlogin* updates */var/adm/acct/sum/loginlog*. This file is evaluated by *prdaily* (see section "prdaily").

USEREXIT
In this state, installation-dependent accounting programs can be invoked under the control of the *runacct.local* shell script.

CLEANUP
In this state, the following actions are taken:
- *prdaily* writes a report on the day's accounting data in */var/adm/acct/sum/rprtMMDD* (see section "prdaily").
- Working files and lock files are deleted.
- *runacct* is terminated.

acctcon
The *acctcon* program is invoked by *runacct* in the CONNECT state. It merges the connect accounting data from the */var/adm/acct/nite/tmpwtmp* file to form total accounting records in */var/adm/acct/nite/ctacct.MMDD*. It also merges user session times in the */var/adm/acct/nite/ctmp* file and writes reports on terminal line usage to */var/adm/acct/nite/lineuse* and reports on accounting routine startup and shutdown times to */var/adm/acct/nite/reboots* . The last two files are evaluated by *prdaily*.

acctprc
The *acctprc* program is invoked by *runacct* in the PROCESS state. It merges the process accounting data from the */var/adm/Spacct?.MMDD* files to form total accounting records in */var/adm/acct/nite/ptacct?.MMDD* files.

acctmerg
The *acctmerg* program is invoked by *runacct* in the MERGE, FEES, DISK and MERGETACCT states:
- In the MERGE state it merges the */var/adm/acct/nite/ctacct.MMDD* and */var/adm/acct/nite/ptacct?.MMDD* files generated by *acctcon* and *acctprc* into the daily accounting file */var/adm/acct/nite/daytacct* .
- In the FEES state it merges the fees recorded in */var/adm/fee* by *chargefee* into the daily accounting file */var/adm/acct/nite/daytacct*.
- In the DISK state it merges the disk accounting data recorded in */var/adm/acct/nite/disktacct* by *dodisk* into the daily accounting file */var/adm/acct/nite/daytacct*.
- In the MERGETACCT state it writes the data collected in */var/adm/acct/nite/daytacct* to the summary file */var/adm/acct/sum/tacct*, which collects data for periodic summaries.

acctcms
The *acctcms* program is invoked by *runacct* a number of times in the CMS state:
- The first time it is invoked, it processes the */var/adm/Spacct?.MMDD* files to produce a command summary in the */var/adm/acct/sum/daycms* daily accounting file.
- The second time it is invoked, it writes the data from the */var/adm/acct/sum/daycms* file to the summary file */var/adm/acct/sum/cms*, which collects data for periodic summaries.
- The third time it is invoked, it writes the data from */var/adm/acct/sum/daycms* and */var/adm/acct/sum/cms* to */var/adm/acct/nite/daycms* and */var/adm/acct/nite/cms* respectively. The last two files are evaluated by *prdaily* .

prdaily
The *prdaily* program (also refer to *acctsh*(1M) in the "System Administrator's Reference Manual" "[7]") is invoked by *runacct* in the CLEANUP state. It writes a report on the day's accounting data in the */var/adm/acct/sum/rprtMMDD* file:
- it reports on daily line usage by evaluating */var/adm/acct/nite/reboots* and */var/adm/acct/nite/lineuse*.
- it reports on daily resource usage on a per-user basis by evaluating */var/adm/acct/nite/daytacct*.
- it reports on daily command usage by evaluating */var/adm/acct/nite/daycms*.
- it reports on command usage since the last time *monacct* was invoked by evaluating */var/adm/acct/nite/cms*.
- it reports on each user's last login date by evaluating */var/adm/acct/sum/loginlog*.

monacct
The *monacct* program (also refer to *acctsh*(1M) in the "System Administrator's Reference Manual" "[7]") is invoked by the *cron* daemon at the time specified in your */var/spool/cron/crontabs/adm* file (see section "Customizing system files"). It generates summaries for a defined accounting period (typically one month) in the */var/adm/acct/fiscal* directory and creates new summary files in the */var/adm/acct/sum* directory. This involves the following actions:

- The */var/adm/acct/sum/tacct* summary file is renamed */var/adm/acct/fiscal/tacctMM* and a new one is created.
- The */var/adm/acct/sum/tacct.MMDD* backup files are deleted.
- The */var/adm/acct/sum/cms* summary file is renamed */var/adm/acct/fiscal/cmsMM* and a new one is created.
- The */var/adm/acct/sum/rprtMMDD* files are deleted.
- The data from the */var/adm/acct/fiscal/tacctMM*, */var/adm/acct/fiscal/cmsMM* and */var/adm/acct/sum/loginlog* files is merged into */var/adm/acct/fiscal/fiscrptMM*.

prtacct
The *prtacct* program (also refer to *acctsh*(1M) in the "System Administrator's Reference Manual" "[7]") can be used to display any accounting file which is in *tacct.h* format (also refer to *acct*(4) in the "System Administrator's Reference Manual" "[7]"). You can identify files in this format by the string *tacct* in the file name.

prctmp
The *prctmp* program (also refer to *acctsh*(1M) in the "System Administrator's Reference Manual" "[7]") can be used to display the */var/adm/acct/nite/ctmp* file which is created by *acctcon* and contains the collected session times, classified by user ID numbers and login names.

acctcom
The *acctcom* user command displays accounting files which are in *acct.h* format (also refer to *acct* (4) in the "System Administrator's Reference Manual" "[7]"). You can identify files in this format by the string *acct* (but not *tacct*) in the file name.

8.8 Accounting files

The files below */var/adm* in the directory hierarchy are used to collect information for the accounting system.

8.8.1 Accounting routine administration files

/var/adm/acct/nite/lock, /var/adm/acct/nite/lock1
These files stop *runacct* being executed more than once at a time.

/var/adm/acct/nite/lastdate
This file contains the date when *runacct* was last called and prevents it being invoked more than once per day.

/var/adm/acct/nite/active
This file contains progress information relating to *runacct* together with warnings and error messages issued while *runacct* is running.

/var/adm/acct/nite/activeMMDD
This file has the same contents as */var/adm/acct/nite/active* and is created if *runacct* aborts as the result of an error.

/var/adm/acct/nite/fd2log
This file contains diagnostics issued while *runacct* is running.

/var/adm/acct/nite/statefile
This file contains the name of the last state completed by *runacct*. This information allows you to restart *runacct* at the appropriate point if the program aborts.

/var/adm/acct/nite/wtmperror
This file contains messages relating to errors occurring while the */var/adm/acct/nite/wtmp.MMDD* file is being checked in the WTMPFIX state.

/var/adm/acct/nite/wtmperrorMMDD
This file is used to back up data from */var/adm/acct/nite/wtmperror* if *runacct* aborts as the result of an error in the WTMPFIX state.

/var/adm/acct/nite/log
This file contains diagnostics issued while *acctcon* is being executed in the CONNECT state.

8.8.2 Connect accounting files

/var/adm/wtmp
This file is used by *acctwtmp*, *closewtmp*, *utmp2wtmp*, *login*, *init*, *ttymon* and *date* to record raw connect accounting data. In the SETUP state the file is renamed to */var/adm/acct/nite/wtmp.MMDD* and a new one is created for subsequent data capture.
As part of your daily duties you should monitor */var/adm/wtmp*. If it grows rapidly, it is advisable to run *acctcon* to determine which terminal line is malfunctioning most. (An extremely high rate of timeouts will adversely affect system performance.) Call *acctcon* as follows and then examine the file specified as the option *-l* argument:

```
# /usr/lib/acct/acctcon -l filename < /var/adm/wtmp 2> /dev/null
```

/var/adm/acct/nite/wtmp.MMDD
This file is the working copy of */var/adm/wtmp*. In the WTMPFIX state it is verified for integrity, corrected if necessary, copied to */var/adm/acct/nite/tmpwtmp* and then, in the CLEANUP state, renamed */var/adm/acct/nite/owtmp*.

/var/adm/acct/nite/owtmp
This file contains the data from the previous day's */var/adm/acct/nite/wtmp.MMDD* file.

/var/adm/acct/nite/tmpwtmp
This file is the corrected version of */var/adm/acct/nite/wtmp.MMDD*. In the CONNECT state, the data in it is merged to form total accounting records in */var/adm/acct/nite/ctacct.MMDD* .

/var/adm/acct/nite/ctacct.MMDD
This file contains the total connect accounting records. In the MERGE state, the records in it are merged with the total process accounting records in the daily accounting file */var/adm/acct/nite/daytacct* .

/var/adm/acct/nite/ctmp
This file contains the collected session times in ASCII format, classified by user ID numbers and login names. It is generated by the *acctcon* program from the data in the */var/adm/acct/nite/tmpwtmp* file, and the *prctmp* program can be used to display its contents.

/var/adm/acct/nite/lineuse
This file contains terminal line usage data. It is generated in the CONNECT state from the data in */var/adm/acct/nite/tmpwtmp* , and it is evaluated by *prdaily*.

/var/adm/acct/nite/reboots
This file contains data relating to accounting routine startup and shutdown times as well as a list of reboots. It is generated in the CONNECT state from the data in */var/adm/acct/nite/tmpwtmp* , and it is evaluated by *prdaily*.

/var/adm/acct/nite/loginlog
This file contains the date and time of each user's last login. It is updated by *lastlogin* in the CMS state, and it is evaluated by *prdaily*.

8.8.3 Process accounting files

/var/adm/pacct
This file is used by the *exit* program to record raw process accounting data. On reaching a defined size (see section "ckpacct") and in the SETUP state, the file is renamed as a */var/adm/pacct?* file, where *?* stands for a sequentially incrementing number. A new */var/adm/pacct* file is then created for subsequent data capture.

/var/adm/pacct?
These files are the copies of each current */var/adm/pacct* file. In the SETUP state they are renamed */var/adm/Spacct?.MMDD* .

/var/adm/Spacct?.MMDD
These files are the working files generated from the */var/adm/pacct?* files. In the PROCESS state the data in them is merged to form total accounting records in the

/var/adm/acct/nite/ptacct?.MMDD files.

/var/adm/acct/nite/ptacct?.MMDD
These files contain the total process accounting records. In the MERGE state the records in them are merged with the total connect accounting records in the daily accounting file */var/adm/acct/nite/daytacct* .

8.8.4 Special services accounting file

/var/adm/fee
This file lists the fees that the *chargefee* program has charged to users. In the FEES state the fees are merged with the total connect and process accounting records in the daily accounting file */var/adm/acct/nite/daytacct*.

8.8.5 Disk accounting files

/var/adm/dtmp
This file is used by *dodisk* to record raw disk accounting data which *acctdisk* merges to form total accounting records in */var/adm/acct/nite/disktacct*.

/var/adm/acct/nite/disktacct
This file contains the total disk accounting records. In the DISK state the records in it are merged with the total connect, process and fee accounting records in the daily accounting file */var/adm/acct/nite/daytacct* .

8.8.6 Summary files for daily accounting

/var/adm/acct/nite/daytacct
This file contains the total connect, process, fee and disk accounting records for one day. It is evaluated by *prdaily* to produce the daily resource usage report. In the MERGETACCT state the records in it are merged into the */var/adm/acct/sum/tacct* summary file for the current accounting period (typically a month).

/var/adm/acct/sum/tacct.MMDD
This file contains the values from the */var/adm/acct/nite/daytacct* file for day *MMDD* . Copies made in the course of an accounting period are deleted by *monacct*.

/var/adm/acct/sum/daycms
This file contains the command summary for one day. It is generated in the CMS state from the values in the */var/adm/Spacct?.MMDD* files. The records in it are merged into the */var/adm/acct/sum/cms* summary file for the current accounting period (typically one month).

/var/adm/acct/nite/daycms
This file is a copy of */var/adm/acct/sum/daycms* and is evaluated by *prdaily* to produce the daily command usage report.

/var/adm/acct/sum/rprtMMDD
This file contains the *prdaily* output for day *MMDD* in ASCII format. This comprises reports on daily line usage, daily resource usage, daily command usage, command usage since the last time *monacct* was called, and each user's last login date. For examples of the reports

contained in this file see section "Accounting reports". Files generated in the course of an accounting period are deleted by *monacct*.

8.8.7 Summary files for the current accounting period

/var/adm/acct/sum/tacct
This file contains the summary of all the total accounting records from the */var/adm/acct/nite/daytacct* daily accounting files for the current accounting period. *monacct* renames this file */var/adm/acct/fiscal/tacctMM* and creates a new one.

/var/adm/acct/sum/cms
This file contains the summary of all the command summaries from the */var/adm/acct/sum/daycms* daily accounting files for the current accounting period. *monacct* renames this file */var/adm/acct/fiscal/cmsMM* and creates a new one.

/var/adm/acct/nite/cms
This file is a copy of */var/adm/acct/sum/cms* and is evaluated by *prdaily* to produce the command usage report for the current accounting period.

8.8.8 Summary files for the current year

/var/adm/acct/fiscal/tacctMM
This file contains the summary of all the total accounting records from the */var/adm/acct/sum/tacct* file for accounting period *MM*.

/var/adm/acct/fiscal/cmsMM
This file contains the summary of all the command summaries from the */var/adm/acct/sum/cms* file for accounting period *MM*.

/var/adm/acct/fiscal/fiscrptMM
This file contains the summaries from the */var/adm/acct/fiscal/tacctMM*, */var/adm/acct/fiscal/cmsMM* and */var/adm/acct/sum/loginlog* files for accounting period *MM* in ASCII format. For the given period it includes information relating to resource usage, command usage, and each user's last login date. Its layout is the same as the */var/adm/acct/sum/rprtMMDD* file created by *prdaily* (see section "Accounting reports").

8.9 runacct error messages

The *runacct* program may abort for a number of reasons, with space problems in the */var* directory and inconsistencies in the *wtmp* file being the commonest causes. If the *activeMMDD* file exists, this is the first place to check for error messages. If the *active*, *lock* and *lock1* files exist, you should check the *fd2log* file for messages. The following is a list of the error messages generated by *runacct* and the recommended recovery actions:

`locks found, run aborted`
: The *lock* and *lock1* files existed when *runacct* was invoked. These files must be deleted before *runacct* can restart. This error may occur if two processes try to start *runacct* at the same time or if *runacct* aborted prematurely and the locks were not deleted before it was called again. Check *fd2log* for error messages to this effect.

`acctg already run for date: check /var/adm/acct/nite/lastdat`e
: The date in the *lastdate* file is the current date, which means that *runacct* has already been run on the same day. If you do want to restart the evaluation, you must delete *lastdate*.

`turnacct switch returned rc=?`
: Check the existence and the execute permissions of the *turnacct* and *accton* programs.

`/var/adm/acct/nite/wtmp.`*`MMDD`* `already exists, run setup manually`
: The *wtmp* file has already been copied to *wtmp.MMDD*.

`wtmpfix errors see /var/adm/acct/nite/wtmperror`
: The *runacct* program has discovered inconsistencies in the *wtmp.MMDD* file in the WTMPFIX state. Use *fwtmp* to eliminate the inconsistencies (see section "Eliminating inconsistencies in files").

`Invalid state, check /var/adm/acct/nite/statefile`
: The *statefile* file is probably defective. Check this file and also the *active* file.

8.10 Restarting runacct

When restarting *runacct* you must specify the date (month and day) for which accounting is to be repeated as an argument in the form *MMDD*.
The point at which processing commences is governed by the contents of the *statefile* file. If you want to override what is in *statefile*, you must either enter the required initial state in *statefile* or specify it on the command line. Two examples are given below.
In the first example, *runacct* is restarted in the state recorded in *statefile*; in the second example the state recorded in *statefile* is overridden:

```
# nohup runacct 0601  2>  /var/adm/acct/nite/fd2log &
# nohup runacct 0601 WTMPFIX  2>  /var/adm/acct/nite/fd2log &
```

8.11 Eliminating inconsistencies in files

It may occasionally happen that a file can no longer be read or gets lost. Many files can simply be disregarded or restored from a backup copy; but with certain files the inconsistencies must be eliminated to maintain the integrity of the accounting system.

8.11.1 Fixing bugs in wtmp files

If the date is changed while the system is in multiuser mode, the date change is recorded in the *wtmp* file. In the WTMPFIX section of the *runacct* program the time stamps of the date changes are adjusted accordingly in the corresponding records (refer to the description of the *wtmpfix* program under *fwtmp*(1M) in the "System Administrator's Reference Manual" "[7]").
What sometimes happens is that date changes coinciding with a reboot are overlooked. As a result, the *acctcon* program crashes in the CONNECT section. To repair a *wtmp* file you can take the following actions:

- Change directories to */var/adm/acct/nite*.
- Create an ASCII version of the *wtmp.MMDD* file:

```
# fwtmp < wtmp.MMDD > filename
```

- In this ASCII version, delete all defective records prior to the date change.
- Overwrite the contents of the *wtmp.MMDD* file with the corrected data:

```
# fwtmp -ic < filename > wtmp.MMDD
```

- Restart *runacct* in the WTMPFIX state.

If *wtmp.MMDD* is irreparable, use *nulladm* to create an empty *wtmp.MMDD* file. This will suppress connect-time accounting for that day. A side effect of the absence of this file is that *acctprc* is prevented from identifying the login to which a particular process belonged, so the process is charged to the owner of the first login with the matching user ID number in the password file.

8.11.2 Fixing bugs in tacct files

It may occasionally happen that records with negative numbers, duplicated user ID numbers or a user ID of 65535 appear in *tacct* files. If so, the first thing to do is check the *sum/tacctprev* file and display it using *prtacct*. If you cannot find any inconsistencies, you should fix the latest *sum/tacct.MMDD* file and create a new *sum/tacct* file. One simple approach to this is as follows:

- Change directories to */var/adm/acct/sum*
- Create an ASCII version of the *tacct.MMDD* file:

```
# acctmerg -v < tacct.MMDD > filename
```

- Delete all defective records from the ASCII version.
- Overwrite the contents of the *tacct.MMDD* file with the corrected data:

```
# acctmerg -i < filename > tacct.MMDD
```

- Create a new *sum/tacct* file:

```
# acctmerg tacctprev < tacct.MMDD > tacct
```

9 Data security and data protection

Another of the system administrator's duties is to guarantee the security of data held on a computer. Security problems are essentially of two types:

- data loss owing to system failure, force majeure, etc.
- data loss or violation of confidentiality due to unauthorized system access

This chapter deals with the problem of unauthorized access and examines the security rules that you should observe. Strategies for preventing system failure are discussed in chapter "High availability".
All system administrators are strongly advised to devote great care and attention to the subject of security and find out more about it from the UNIX literature. All we can do here is provide a few basic observations and tips.

Note:

For detailed information on the subject of data security and protection refer to sources such as the "UNIX System Security" manual "[26]".

9.1 AUDIT and ASECO

Siemens Nixdorf offers two professional security systems, AUDIT and ASECO.
The AUDIT program package detects and records security-related user actions. Its records contain information identifying the user who initiated the action, the date and time when it took place, whether it succeeded, and the name, type, device, inode and file system of the objects involved.
ASECO replaces password-based access control with a mechanism based on the use of chipcards (smartcards), and it features a double verification of identity. There is a secret personal identification number stored on the card, and the system prompts for it and checks it. In addition, the card and a security module on the system have an identical secret key for message encoding. A random number is encrypted in the card and in the security module, and the results are compared by the security module. Access is not granted unless both of these security checks are passed.
Ask for information about AUDIT and ASECO at your local Siemens Nixdorf office.

9.2 System security

The subject of system security can be broken down into a technical aspect and an organizational aspect. The organizational aspect relates to the responsibilities, the privileges and the trustworthiness of those who have access to the system. This, however, is not primarily a subject for the system administrator and so is not dealt with here. The technical aspect which is dealt with below can be further classified as hardware security and software security.

9.2.1 Hardware security

Protecting hardware is just as important as protecting software. Of particular importance is where your hardware is physically located:

- The console monitor and the system unit should be in a lockable room and should under no circumstances be freely accessible.
  The reasons for this are obvious: someone could switch off or damage the system, which might result in data loss. A small computer could be stolen. Furthermore, someone with access to the operating system installation media could break into the system.
- Also make sure that the system is protected against fire, water, power failure and power surges. Actions designed to prevent system failure are described in chapter "High availability".

9.2.2 Software security

Both the operating system and the applications running on it must minimize the risk of unauthorized access. As a rule, you as system administrator cannot guarantee the security of applications; that is up to the programmers. What you can do is ensure that the system is configured in such a way as to offer no security loopholes. This particularly relates to the protection of user accounts and to the allocation of file and directory permissions.

- All accounts must be password-protected. Make absolutely sure that not only *root* but all other administrative accounts such as *uucp* , *daemon*, *sys* and *install* are password-protected.
- Reliant UNIX checks no more than the first eight characters of a password. Therefore the passwords which are used should be at least eight characters long, include special characters and use a mixture of uppercase and lowercase letters. You can set a minimum length for passwords by defining an appropriate value for the *PASSLENGTH* variable in the */etc/default/passwd* file.
- Users should not choose passwords which can be derived or deduced from personal associations (their name, their dog's name and so on).
- To ensure that users change their passwords regularly you can enable the password aging mechanism. Then a password will become invalid after a defined period and the user will have to change it (see section "Protecting and locking accounts").
- You should never delete a password with the command *passwd -d login-name*, as this makes the account accessible without a password check and creates a security loophole.
- The system should be configured to allow *root* logins only at the console (refer to the */etc/default/login* file in section "Files in /etc/default").
- If you have logged in as *root* or have used the *su* command to give yourself superuser privileges, you should always log out if you leave your terminal unattended.

- Use the *root* account to perform system administration duties only.
- Never use the *su root* command when working under another user's login name. An experienced user will know how to log the *root* password when you enter it.
- Any account which is no longer being used and is therefore no longer required represents an increased security risk. Such accounts should be deleted.
- If at all possible, you should not set up temporary accounts, i.e. accounts on which various users work on a temporary basis, because such accounts cannot be uniquely associated with one user. If you do use temporary accounts on your system, you should release them for use for short periods only.
- Be sure to allocate a separate user ID number (UID) to each account. If there are two users working under the same user ID, each has access to the files the other creates.
- Directories with write permission for all users are a security risk. At regular intervals you should check the permissions on particularly sensitive directories such as */etc*, */sbin* and */usr/sbin*.
  Write permission for all users should also not be allocated to login directories, users' *.profile* files or log files (see section "Log files").
- Instruct your users to regularly check their *$HOME/.rhosts* file. The login names listed in this file are granted access without a password check to the account on which the *.rhosts* file is set up. The *$HOME/.rhosts* file must have read and write permissions for the owner only.
- Make sure that users of graphical interfaces do not use the command *xhost +* to bypass all access restrictions for their local displays.

9.2.3 Setting up a secure environment

To bar access to sensitive data and largely deny any opportunities to break into the operating system, you can set up a "protected subsystem" on your computer. Depending on your security requirements, you can even configure this subsystem such that no users can work anywhere else. However, it will generally be sufficient to set up a protected subsystem only for specific accounts such as temporary accounts.
To set up a protected subsystem you first have to enter an asterisk (*) in the seventh and last column of the */etc/passwd* file entry for the account. This column defines the startup program which is run when the user logs in (see section "Format and function of the /etc/passwd file"). If it contains an asterisk, the directory specified in the sixth column is defined as the user's root directory. The effect of this is that the user will not be able to gain access to higher levels in the directory hierarchy.
To allow a user trapped in this subsystem to do any work at all, though, you need to supply the subsystem with directories and files which provide a usable but restricted system environment. Among other things this entails defining which commands are available to the subsystem user.

Example of a protected subsystem

This example of a protected subsystem on Reliant UNIX 5.4x is based on the following entry in the */etc/passwd* file:

```
# grep guest /etc/passwd
guest:x:555:555::/home/subsystem:*
#
```

The */home/subsystem* directory is the new root directory for the *guest* account. Below */home/subsystem*, which must not have write permission for the group or for others, you must

now set up at least the following directories: *bin*, *dev*, *etc*, *home*, *sbin*, *usr* and *var*.

```
# cd /home/subsystem
# ls -l
total 14
lrwxrwxrwx  1 root    other     4096 Feb 10 13:51 bin -> /usr/bin
drwxr-xr-x  2 root    other     4096 Feb 10 13:51 dev
drwxr-xr-x  3 root    other     3584 Mar 4  07:07 etc
drwxr-xr-x  3 root    other      512 Feb 24 13:26 home
drwxr-xr-x  2 root    other     2048 Dec 2  09:31 sbin
drwxr-xr-x  4 root    other     1024 Feb 10 13:09 usr
drwxr-xr-x  3 root    other      512 Feb 10 13:44 var
#
```

These directories contain the following files and subdirectories:

```
# cd /home/subsystem/dev
# ls -l
total 0
crw-r--r--  1 root    other    16, 0 Feb 10  1994 tty
crw-r--r--  1 root    other     2, 4 Feb 10  1994 zero
#
# cd /home/subsystem/etc
# ls -l
total 14
-rw-r--r--  1 root    other      438 Feb 10  1994 group
-rw-r--r--  1 root    other     2124 Feb 10  1994 passwd
-rw-------  1 root    other     1638 Feb 10  1994 shadow
#
```

The *group*, *passwd* and *shadow* files must contain only the essential entries. The contents of the *passwd* file should be something like this:

```
# cat passwd
root:x:0:1:0000-Admin(0000):/:
daemon:x:1:1:0000-Admin(0000):/:
bin:x:2:2:0000-Admin(0000):/usr/bin:
guest:x:555:555::/home/guest:/sbin/sh
#
# cd /home/subsystem/home
# ls -l
total 2
drwxr-xr-x  3 guest    guest      512 Feb 24 13:26 guest
#
# cd /home/subsystem/sbin
# ls -l
total 288
-rwxr-xr-x  1 root    other  133040 Feb 24 13:26 sh
#
# cd /home/subsystem/usr
# ls -l
total 4
drwxr-xr-x  2 root    other     512 Feb 24 13:26 bin
drwxr-xr-x  2 root    other     512 Feb 24 13:28 lib
#
```

In the following example, user *guest* is granted access only to a very restricted set of commands: only the *ced* editor is made available.

```
# cd /home/subsystem/usr/bin
```

```
# ls -l
total 192
-r-sr-xr-x  2 root    other   49468 Feb 24 1993 login
-r-xr-xr-x  2 root    other  128432 Feb 24 1993 ced
#
# cd /home/subsystem/usr/lib
# ls -l
total 640
-r-xr-xr-x  2 root    other  318880 Feb 24 1993 libc.so.1
#
# cd /home/subsystem/var
# ls -l
total 2
drwxr-xr-x  2 root    other     512 Feb 24 13:26 adm
#
# cd /home/subsystem/var/adm
# ls -l
total 758
-r--r--r--  2 root    other     512 Feb 24 13:26 lastlog
-rw-r--r--  2 root    other     512 Feb 24 13:28 utmp
-rw-r--r--  2 root    other     512 Feb 24 13:28 utmpx
-rw-r--r--  2 root    other     512 Feb 24 13:28 wtmp
-rw-r--r--  2 root    other     512 Feb 24 13:28 wtmpx
#
```

Restricted shell

In addition, you can supply selected users with only a restricted shell as their startup shell. This allows you to define which commands a user is allowed to run and which directory a user is allowed to work in. For an explanation of the functions of a restricted shell refer to the description of *sh* in the manual "Commands" "[5]".

10 High availability

This chapter deals with the second aspect of system security: high availability. The objective of high availability strategies is to provide protection against data loss in the event of system failures. The underlying principle is to keep the system operating even after possible malfunctions, thereby enhancing system availability.
High availability is not a built-in system capability, as is the case with fault-tolerant systems, but is based on system enhancements which are extensions of Reliant UNIX standards. Thus high availability can be achieved in stages by adding various components:

- The first stage of high availability is "data availability". Components which help guarantee data availability include user-friendly backup programs, parallel data storage (e.g. disk mirroring, see section "Disk mirroring") and transaction backups (where a number of interrelated backup actions are treated as a unit).
- The second stage is "device availability", characterized by redundancy among system components (power supplies, controllers, disks and so forth).
- The third stage, "system availability" or full high availability, requires a configuration featuring at least two computers.

The actions the system administrator can take in relation to high availability are initially more restricted in scope than is the case with data security, as system enhancements are involved. You should, however, give thought to high availability and use your influence if you consider that high availability enhancements are required.

Note:

For detailed information on the subject of high availability refer to the "High Availability Guide" "[27]".

10.1 RAID strategies

RAID technology (Redundant Array of Independent Disks) is based on the grouping of a number of hard disk drives to form disk arrays. Part of the physical storage capacity in a disk array is used to store redundant information about data. If one disk drive in an array fails, the missing data can be reconstructed.
RAID implementations are assigned to levels from 1 to 5, as explained below.

10.1.1 RAID 1

RAID 1 is also known as disk mirroring. With this approach, data is written to two separate drives which can each be read independently. RAID 1 is thus particularly well suited to applications which demand very high availability. For details of the disk mirroring strategy refer to section "Disk mirroring".

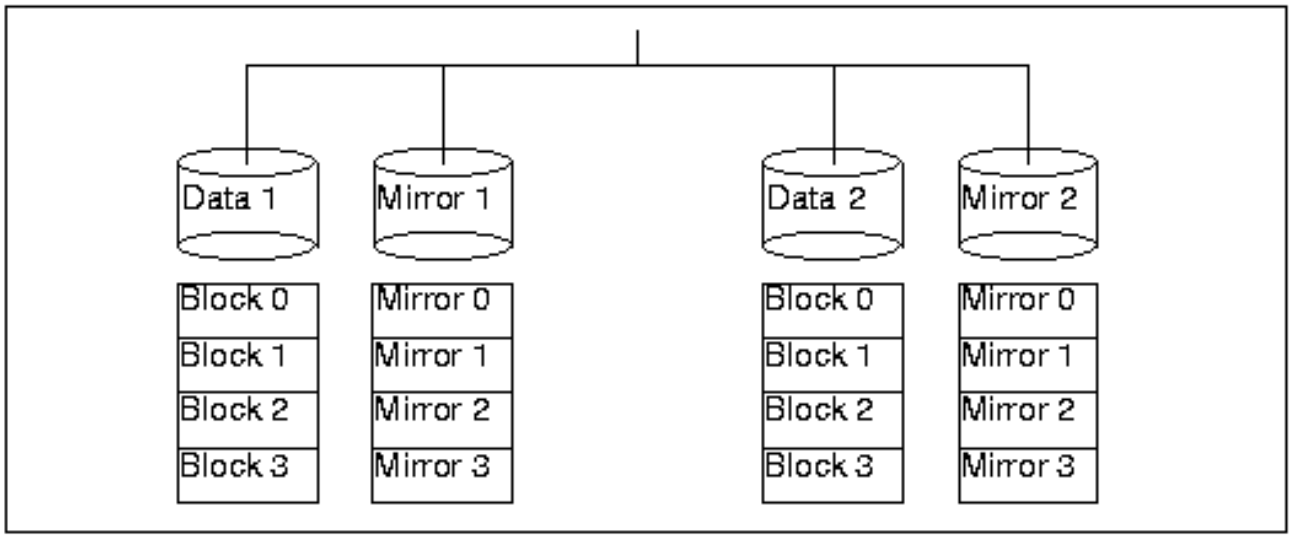

Figure 16: RAID level 1

10.1.2 RAID 2

On RAID level 2, data blocks are distributed among multiple parallel drives. A complex checksum technique (Hamming code) is applied to produce check bits which guarantee that data can be restored after system failures.
This RAID configuration does, however, require multiple hard disks for the checksums. A further disadvantage is that even for small data transfers all the drives in the group must be accessed. For these reasons this technique is rarely used nowadays.

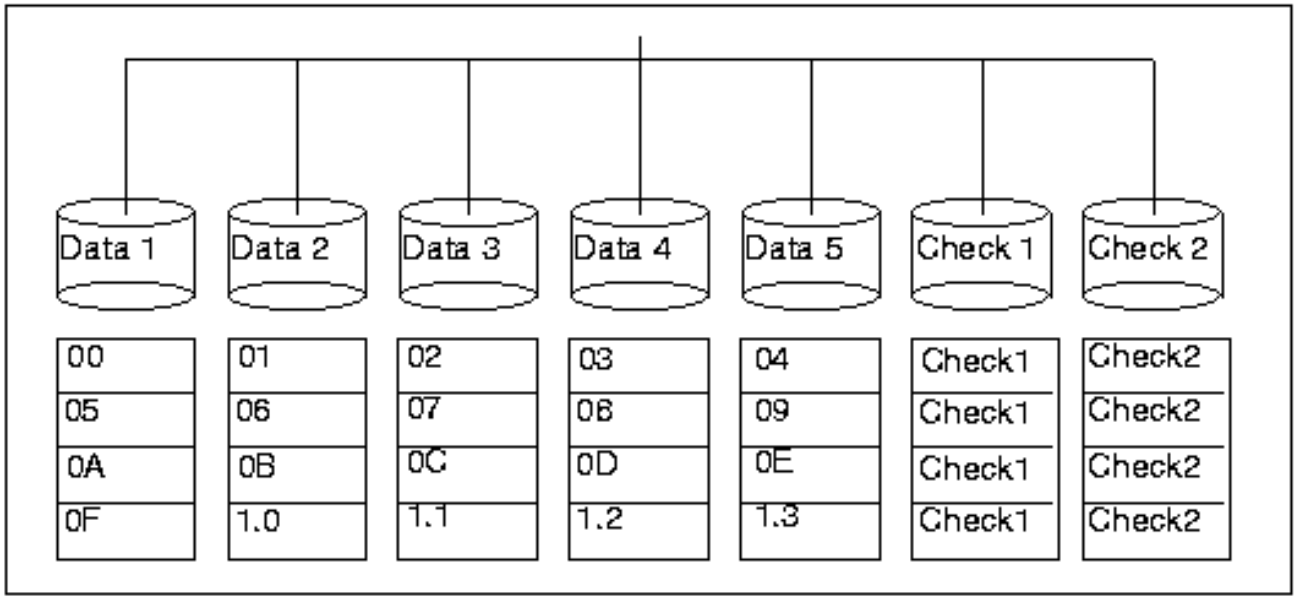

Figure 17: RAID level 2

10.1.3 RAID 3

Data is distributed among multiple drives of the same size (striping), with a single parity disk for each group of drives.
Reads and writes are performed in parallel across the entire group, which means that all the drives within a group are active in a single read or write operation. For that reason RAID level 3 is likewise unsuitable for low-volume data transfers.

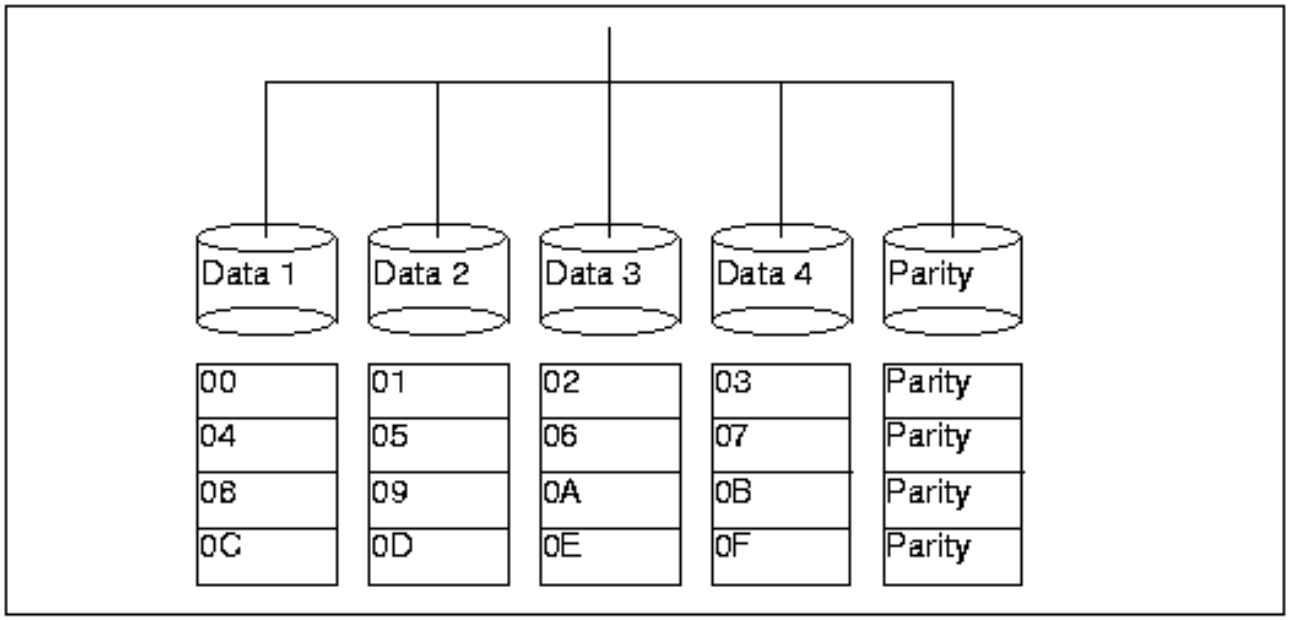

Figure 18: RAID level 3

10.1.4 RAID 4

Data is distributed sector by sector in system blocks among multiple drives in series. A parity block corresponding to these data blocks is set up and stored at the same logical block address on the parity drive.
Thus on RAID level 4, each time a drive is accessed, a parity block on the parity disk has to be updated. As there is only one parity disk, a bottleneck may arise.

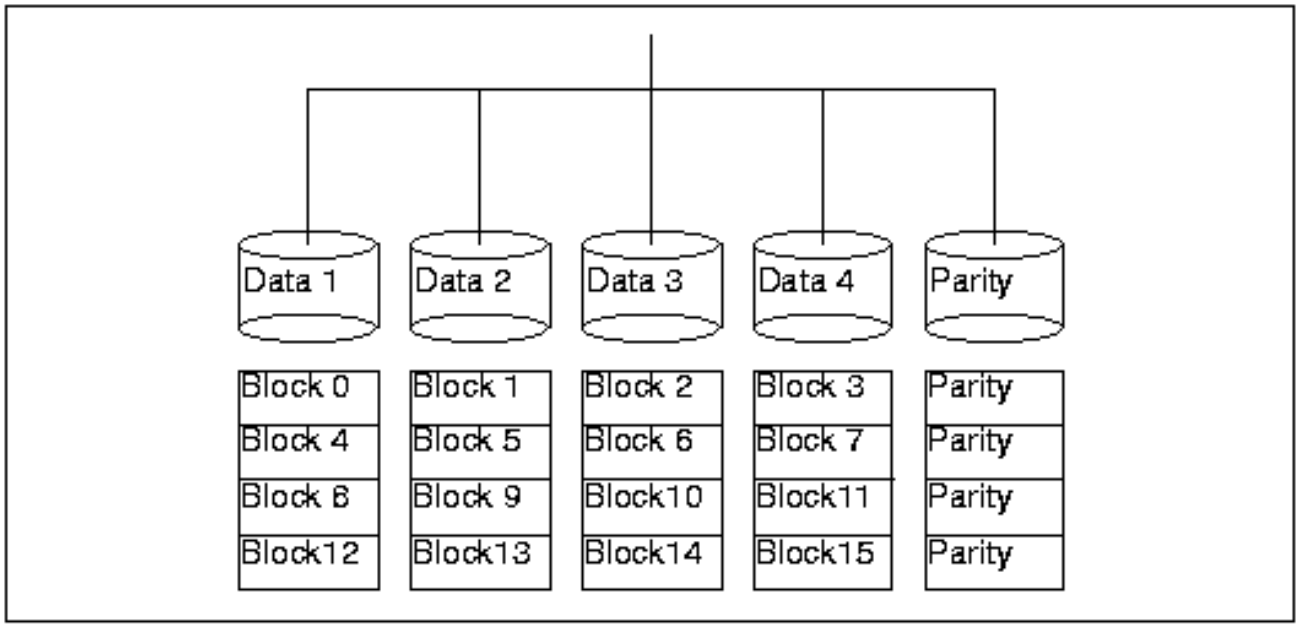

Figure 19: RAID level 4

10.1.5 RAID 5

Data is stored sector by sector in system blocks across multiple drives. A parity block matching these data blocks is formed and is evenly distributed among all the drives in a spiral pattern.
Thus the parity information is not stored on a single drive, as with RAID 3 and RAID 4, but is evenly distributed among all the disks in the group.
RAID level 5 is suitable for both low-volume and high-volume data transfers and satisfies the requirements of high availability and disk use.

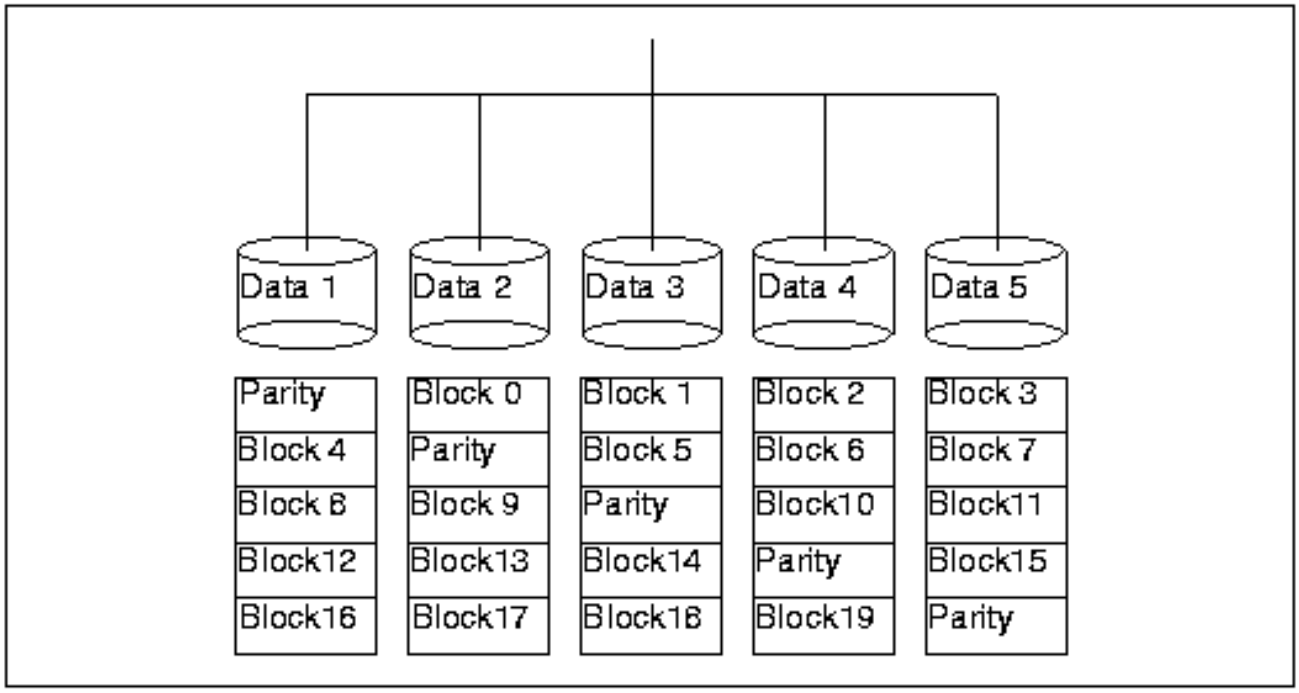

Figure 20: RAID level 5

10.1.6 RAID level summary

Of the RAID levels described above only RAID 1 and RAID 5 are of importance in practice. Levels 2, 3 and 4 are rarely used today.
RAID 1 is particularly suitable for environments in which performance is critical and is the only possible RAID configuration if no more than two drives are used.
RAID 5 is used in multi-user environments which either do not require high performance or have few write operations. At least three drives are required for RAID 5 arrays.

10.1.7 RAID configurations with RAIDmaster controllers

Siemens Nixdorf offers high-performance PCI RAID controllers. With these "RAIDmaster controllers" you can conveniently create and administer RAID configurations tailored to your individual requirements via the "Storage Manager" user interface.
The following list summarizes the advantages of the RAIDmaster controllers. For more information refer to the manual "RAIDmaster, System Administrator's Guide" "[33]".

- RAID 1 and RAID 5 are supported
- The array groups can be automatically protected from a hard disk failure by means of additional drives (hot spares). In the event of a drive of an array group failing, a hot spare is automatically integrated in the array group and takes on the functions of the failed drive.
- Due to the hardware design and the controller features, drives can be replaced online.
- All events in the controller/disk subsystem are logged and can be sent off via e-mail.
- The voltage and temperature on the controllers is monitored and can be polled.
- Failed drives are signaled by LEDs at the disks.
- A controller cache of up to 64 Mbytes is used for quickly buffering jobs from and to the disks, independently of the operating system.

- A batch configuration of several identical RAID groups can be easily configured.
- Parallel assignment of SCSI jobs to all disks allows I/O overlapping and Tagged Command Queuing.
- In addition to the configuration, diagnosis and I/O monitoring can also be performed locally.

10.2 Disk mirroring

Reliant UNIX features a hard disk management mechanism which allows for the configuration of "virtual disks". A virtual disk operates in the same way as a conventional physical disk, but it is formed by combining parts of physical disks to form a virtual hard disk partition. One example is the technique known as disk mirroring.
Disk mirroring (RAID level 1) is implemented using virtual disks, with all I/O operations being performed on two or more physical devices simultaneously. That means that the data is available in at least two copies on various hard disks. Thus disk mirroring enhances data availability.

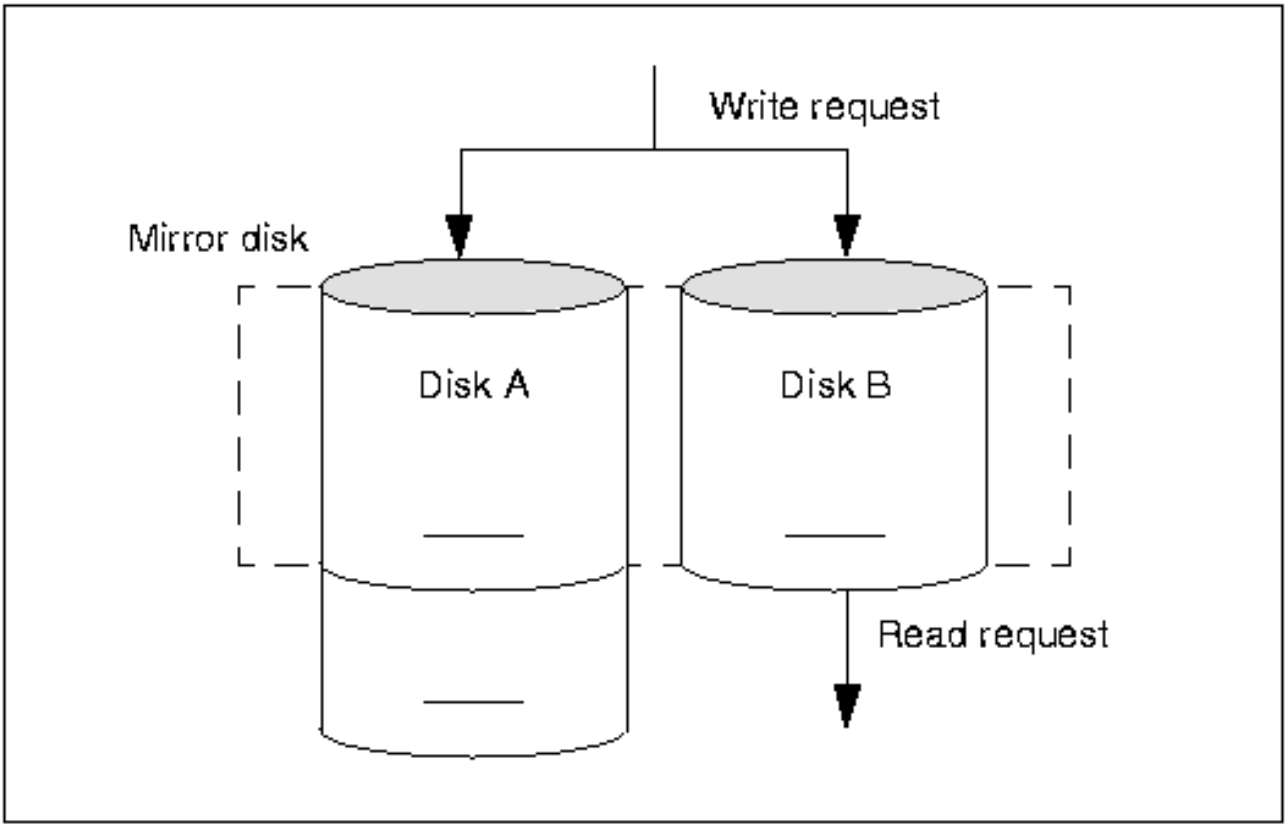

Figure 21: Example of a mirrored disk configuration

A write access to the mirror disk is converted to write accesses to each of its components. This ensures that each part contains exactly the same data.
A read access is converted to a read access to just one of the parts. As all parts contain exactly the same data, the part the data can be read from fastest is used. This results in a considerable increase in the speed with which reads are performed, especially in conjunction with memory disks (virtual disks which use Reliant UNIX system memory as a data storage medium).
In the event of an unrecoverable or severe error in a read/write access, the part in which the defect occurred is disabled. As long as the read/write access can be performed using another part, the access is considered successful from the user's viewpoint. As the user process can continue functioning in the event of isolated malfunctions, the risk of data loss is greatly reduced.
Data remains available while the malfunction is being dealt with. The system administrator

can then bring the disabled part of the mirror disk back on-line.
A mirror disk may consist of two or more parts. In practice mirror disks with up to three parts are customary. No distinction is made between master and slave parts: all parts of a mirror disk are of equal rank.

Note:

For a detailed description of disk mirroring refer to the "Virtual Disks" manual "[12]".

10.3 OBSERVE

OBSERVE is a product designed to monitor multiple-computer configurations. In the event of a system failure, OBSERVE initiates predefined actions. This automated control mechanism keeps failure response times to an absolute minimum.
If one system fails completely, its entire functionality can be supplied by a standby computer. Thus OBSERVE can be classified as a product supporting system availability.
There are essentially two sides to the functionality of OBSERVE:

- OBSERVE primarily monitors the partner system or systems by remotely communicating with them.
- Programs known as observers monitor other hardware and software components. OBSERVE supplies standard observers and also offers a C interface for programming custom observers. Components such as hard disks, controllers, network connections and applications are likely candidates for custom observers.

External disks have to be used to ensure that all data is available to OBSERVE after the failure of one system. When one system fails, OBSERVE switches the external disks over to the system which is still functioning. Conversely the use of disk mirroring is not absolutely essential but does increase availability (OBSERVE enhances the availability of systems; disk mirroring enhances the availability of data).

Note:

For detailed information on OBSERVE refer to the version-specific OBSERVE documentation, such as the manual "OBSERVE V1.2, User Guide" "[28]".

10.4 RMS (Reliant Monitor Software)

RMS (Reliant Monitor Software) is software for ensuring high availability in a "cluster" of computers.
RMS monitors the system components within a computer cluster using so-called "detectors". When a component fails, this triggers a user-defined reaction, e.g. the transfer of an application to another computer in the cluster.
In this way, RMS ensures maximum availability of applications, while it is not responsible for ensuring uninterrupted availability of individual systems.
A cluster must comprise at least two computers, in which case the application data must be physically accessible by some or all the systems in the cluster.

Note:

For more informationen on RMS refer to the manual "RMS V3.1, RMS konfigurieren und administrieren" "[36]".

10.5 SIDLM

The SIDLM (SINIX Distributed Lock Manager) ensures high availability in systems using an ORACLE database.
Under SIDLM, computers sharing the data of an ORACLE database are grouped together to form a multi-computer system ("cluster"). Different database applications may run on each of the computers in the cluster. Database accesses are coordinated, ensuring the consistency of the data.
When a computer fails, the pending transactions are written to disk and the lock entries of the failed computers are canceled. The entire database is still available to the other computers in the cluster.

Note:

For more information on SIDLM refer to the manual "SIDLM V1.2, Installation and Administration Guide" "[37]".

10.6 UPS

An uninterruptible power supply (UPS) is connected between the computer system and the power source and guards the system against brief power interruptions, voltage dips and voltage surges. A UPS can maintain the supply of power for a certain time, depending on the load, the state of its battery and the type of UPS. There is then generally a graceful shutdown. A UPS enhances device availability.
In high-availability configurations, every system component with its own power connection should by protected by its own UPS. The type of UPS used should be matched to the power requirements of the individual components.

Note:

For detailed information on UPS functionality and specifications refer to the documentation accompanying your UPS.

10.7 BBU

A battery backup unit (BBU) protects the system against short power interruptions. A BBU can be an internal component or can be attached to the system externally in the form of a BBU cabinet. A BBU can maintain the supply of power for certain time, depending on the load, the state of its battery and the type of BBU. There is then generally a graceful shutdown. A BBU enhances device availability.
Unlike a UPS, a BBU does not also provide protection for peripherals.

Note:

For detailed information on BBU functionality and specifications refer to the Operating Manual for your system.

11 Related publications

The manuals can be ordered from your local Siemens Nixdorf office.
Some of the more frequently required manuals are also available online. The online documentation is available on CD and can also be accessed on the World Wide Web from Siemens Nixdorf's homepage.

[1] Reliant UNIX
Operating Manual (hardware-specific)
Target group
System administrators, users
Contents
This manual describes how the hardware is installed, and how the system is placed in service and operated. The contents of the manual are dependent on the computer type.

[2] Reliant UNIX 5.44
Reliant UNIX Installation and Operation, RM200, RM300, RM400
Installation Guide
Target group
System administrators
Contents
This manual describes how to install the Reliant UNIX 5.44 operating system both as a new installation and as an update. Also documented are the basics of the operating system and key aspects of using it as well as instructions for installing the software using the SYSADM operator system.

[3] Reliant UNIX 5.44
Reliant UNIX Installation and Operation, RM600-xxx, RM600-E
Installation Guide
Target group
System administrators
Contents
This manual describes how to install the Reliant UNIX 5.44 operating system both as a new installation and as an update. Also documented are the basics of the operating system and key aspects of using it as well as instructions for installing the software using the SYSADM operator system.

[4] Reliant UNIX 5.44
Documentation Guide
Annotated Overview, Index
Target group
Users and system administrators, network users and administrators, Spool users and administrators and programmers.
Contents
The "Documentation Guide" provides an overview of the documentation available for Reliant UNIX as well as the titles of the manuals and their order numbers. As well as a listing of the contents of the most important manuals, an index is provided which sets out the activities covered by the respective manuals and acts as a reference to the manual where the actual activity is documented.

[5] Reliant UNIX 5.44
Commands, User Reference Manual
Reference Manual
Target group
Users, system administrators
Contents
This manual is intended for reference purposes. It describes the user commands of the Reliant UNIX 5.44 operating system.

[6] Reliant UNIX 5.44
User's Guide
User Guide
Target group
Users
Contents
The "Users Guide" describes the main elements of the Reliant UNIX operating system, including an introduction to using SINIX, the file system, process handling and the shell.

[7] Reliant UNIX 5.44
System Administrator's Reference Manual
Reference Manual
Target group
System administrators
Contents
This manual describes the system administrator commands, file formats, device files and procedures for system maintenance in alphabetical order.

[8] Reliant UNIX 5.44
Network Administration
System Administrator's Guide
Target group
Network administrators managing UNIX systems in TCP/IP networks
Contents
This manual describes the network administration activities that have to be performed when using the TCP/IP software on Reliant UNIX 5.44 as well as the basic network function (BNU).

[9] Reliant UNIX 5.44
Network Reference Manual
Reference Manual
Target group
Network administrators, system administrators
Contents
This manual gives a description in alphabetical order of the commands, C functions, file formats, system maintenance procedures as well as other Reliant UNIX tools for operating networks.

[10] Reliant X 5.44
System Administration and Hardware Configuration Using the SYSADM User Interface
System Administrator's Guide
Target group

System administrators
Contents
This manual describes the SYSADM user interface with the integrated configuration tool Config. The Config is available under the menu option Configuration once the operating system has been installed in SYSADM. The user interface provides a user-friendly means of configuring and managing I/O devices.

[11] Reliant UNIX 5.44
Hardware Configuration with Config under SINIX/windows
Target group
System administrators
Contents
This manual describes the functionality of Config under the SINIX/windows user interface.

[12] Reliant UNIX 5.44
Virtual Disks
User Guide
Target group
System administrators
Contents
This manual describes the various types of virtual disk and how they are configured.

[13] Reliant UNIX 5.44
Veritas File System (VxFS) V1.2
System Administrator's Guide
Target group
System administrators
Contents
This manual provides an overview of the structure and the features of the Veritas file system (VxFS) and also describes how it is used. In addition, the commands and utilities used with VxFS are documented and detailed explanations of error and diagnostic messages are given.

[14] Reliant UNIX 5.44
Tuning Guide
System Administrator's Guide
Target group
System administrators
Contents
This manual describes options for improving the system performance.

[15] Xprint V5.0
User and Administrator Guide
Target group
Reliant UNIX users and system administrators; Spool administrators
Contents
This manual provides information on the Spool concept and documents the objects that make up the Spool. In addition, it describes how Spool is installed as well as how it is used via the command interface.

[16] Xprint V5.0
Reference Manual
Target group

Reliant UNIX users and system administrators; Spool administrators
Contents
This manual describes the Spool commands in alphabetical order, and documents the Spool messages, the configuration files for Spool objects as well as the standard Spool data formats. The manual also presents an overview of how the Spool system works.

[17] SINIX
AT&T Spool
System Administrator's Guide
Target group
System administrators
Contents
This manual describes how to set up and manage the LP print service.

[18] SINIX/windows User Environment V3.1
Documentation Overview
Target group
SINIX/windows users
Contents
The manual provides an overview of the SINIX/Windows documentation.

[19] SINIX/windows User Environment V3.0 (CDE)
User's Guide
User Guide
Target group
Users with basic knowledge of Reliant UNIX
Contents
This manual describes the underlying features of the SINIX/windows desktop and explains how to work with the desktop and the desktop applications.

[20] SINIX/windows User Environment V3.0 (SINIX Desktop)
Introduction to Handling and Configuration
User Guide
Target group
Users with basic knowledge of Reliant UNIX
Contents
This manual provides an introduction to operating and configuring the SINIX/windows graphical user interface. It describes the components of the SINIX/windows desktop, how to use the user interface, and interactive configuration of the desktop.

[21] SINIX/windows User Environment V3.0 (CDE)
Advanced User and System Administrator Guide
System Administrator's Guide
Target group
System administrators
Contents
This manual covers advanced tasks in customizing the appearance and behavior of the Desktop.

[22] SINIX/windows User Environment V3.1 (CDE)
CDE Enhancements
User Guide
Target group

Users and system administrators
Contents
The book provides an overview of the additional functionality that Siemens Nixdorf Informationssysteme AG an Triteal Corporation have added to the Common Desktop Environment (CDE).

[23] SINIX/windows User Environment V3.0 (SINIX Desktop)
Guide for Experts and System Administrators
System Administrator's Guide
Target group
System administrators
Contents
This manual discusses the concepts underlying SINIX/windows, such as the client-server model, and configuration of a client using resources. It also describes the primary clients.

[24] SINIX/windows User Environment
Clients Reference Manual
Reference Manual
Target group
System administrators
Contents
This manual provides the experienced user with a comprehensive overview of the SINIX/windows clients and the resources used by the clients.

[25] SINIX/windows User Environment
XDCL Desktop Configuration Language
Reference Manual
Target group
System administrators
Contents
This manual provides an overview of the XDCL language,with which the experience user can configure the SINIX/windows desktop.

[26] SINIX
UNIX System Security
User Guide
Target group
Users, system administrators, system programmers
Contents
This manual describes the user classes of a UNIX system, provides an overview of the security risks in a UNIX system, and explains the ways of reducing these risks.

[27] SINIX
High Availability Guide
User Guide
Target group
System administrators, Systemsicherheits-Beauftragte
Contents
The High Availability Guide describes ways of increaing the availability of UNIX systems.

[28] SINIX
OBSERVE

User Guide
Target group
OBSERVE system administrators and OBSERVE programmers
Contents
This manual describes how OBSERVE is installed and configured in single-, dual- and multi-computer configurations.

[29] Reliant UNIX
AUDIT V2.0
System Administrator Guide
Target group
System administrators
Contents
This manual describes how AUDIT is installed and configured.

[30] Reliant UNIX 5.44
UFS UNIX File System
Target group
System administrators
Contents
This manual describes the organization, administration, maintenance and consistency checking for the ufs file system.

[31] Reliant UNIX 5.44
Error Diagnosis and Error Handling
Target group
System administrators, programmers and service technicians
Contents
This manual describes potential system errors, how these errors are interpreted, as well as possible measures for dealing with errors. In addition, the user interfaces available for system diagnostics are described.

[32] IOCS V3.0
Administration, Configuration and Programming of Printers
Target group
System administrators, application programmers
Contents
This manual describes how to manage, configure, install and program printers using IOCS and also outlines the compatible control sequences for printer programming. It provides information on programming with the IOCS but does not include programming information.

[33] Reliant UNIX, Windows NT
RAIDmaster
System Administrator's Guide
Target group
System administrators
Contents
The manual describes the configuration of the RAIDmaster-controllers.

[34] RMS V3.1
RMS Installation
Target group
System administrators

Contents
This manual describes how to install the Reliant Monitor Software (RMS).

[35] RMS V3.1
RMS Operation
Target group
System administrators
Contents
This manual describes the Reliant Cluster Visual Manager (RCVM), the graphical user interface for RMS.

[36] RMS V3.1
RMS Configuration and Administration
Target group
System administrators
Contents
This manual decribes the configuration and administration of RMS. The manual also presents an overview of RMS, gives examples how to configure RMS and how to administer RMS via the command interface.

[37] SIDLM V1.2
Installation and Administration Guide
Target group
System administrators
Contents
The manual decribes how to install the SINIX Distributed Lock Manager (SIDLM) . Also documented is the administration of a SIDLM cluster.

Index