Digital UNIX

System Administration

Part Number: AA-PS2RE-TE

December 1996

Product Version: Digital UNIX Version 4.0B or higher

This guide describes the tasks you perform in order to maintain a Digital UNIX operating system running on an Alpha workstation or server. You use UNIX commands, scripts, and the SysMan graphical user interfaces to perform the system administration tasks described in this manual.

Digital Equipment Corporation
Maynard, Massachusetts

Contents

5 Configuring the Kernel

6 Administering Devices with Dynamic Device Recognition

7 Administering the UNIX File System

8 Administering the POLYCENTER Advanced File System

9 Administering the Logical Storage Manager

11 Administering the Print Services

14 Administering Events and Errors

Index

Figures

Tables

About This Guide

This manual describes the tasks you perform in order to administer a Digital UNIX operating system running on an Alpha workstation or server.

Audience

This guide is intended for system administrators. Administrators should have knowledge of the operating system concepts and commands, and the hardware and software configuration.

New and Changed Features

This revision of the manual documents the following new features, changed features, and retiring interfaces.

Recognizing the Common Desktop Environment GUI

With this release of the operating system, the Common Desktop Environment (CDE) becomes the preferred operating system interface and the *SysMan* graphical user interface, which runs under the CDE, becomes the preferred system administration tool. See Chapter 2 for more information about the interface and its commands.

New Information

This manual includes discussions of the following new software components:

- Chapter 2, System Administration Tools and Methods, which describes the SysMan GUI and Remote Administration
- SCSI Device Dynamic Device Recognition (DDR). See Chapter 6.
- Security. See Chapter 4.
- Bootable Tape. See Chapter 12.
- Telnet (TCP/IP) printing. See Chapter 11.
- DECEvent error logging facility and Environmental (thermal) Monitoring. See Chapter 14.
- Power management for reducing energy consumption. See Chapter 4.

- Performance monitors. See Chapter 4.
- Administering Specific Hardware Devices. See Appendix E

The following chapters have been heavily revised to document new features and to correct documentation errors:

- Chapter 5, Configuring the Kernel
- Chapter 7, Administering the UNIX File System
- Chapter 8, Administering the POLYCENTER Advanced File System
- Chapter 9, Administering the Logical Storage Manager
- Chapter 10, Administering User Accounts and Groups
- Chapter 11, Administering Print Services

Changed Information

The chapter on using the `setld` utility to install and manage software has been moved from this manual to the *Installation Guide*.

The information about adding third party SCSI devices has been replaced with Dynamic Device Recognition (DDR), which performs the same functions. DDR is described in Chapter 6.

The Logical Volume Manager (LVM) appendix has been removed from the manual because the LVM functionality has been retired from the operating system.

Retiring Software

The `uerf` error logging software will be retired in a future release of the operating system. Information about it has been moved to Appendix D.

Support for the Logical Volume Manager (LVM) has been retired in this release of the operating system. All volume management functionality now is provided by the Logical Storage Manager (LSM) as described in Chapter 9. All LVM functionality has been disabled with the exception of the support necessary to encapsulate LVM volumes under LSM. All current users of LVM are now required to encapsulate their LVM volumes under LSM in order to maintain access to their data.

Note

In a future release of the operating system, all support for LVM will be dropped and any data remaining under control of LVM software will be lost.

For more information about LVM volume encapsulation, refer to the *Logical Storage Manager Reference Guide.*

Information about LVM has been deleted from this manual.

Unchanged Information

With the exception of minor documentation problem fixes, the information in the following chapters and appendixes has not changed since the last version of the manual:

- Chapter 3, Starting Up and Shutting Down the System, which combines the former chapters about startup and shutdown.
- Chapter 12, Administering the Archiving Services
- Chapter 13, Administering the System Accounting Services
- Appendix A, Device Mnemonics
- Appendix B, SCSI/CAM Utility Program
- Appendix C, Support of the CI and HSC Hardware

Organizational Changes

Chapter 1 has been expanded and most of the chapters have been renamed. Scan the following section for more information.

Organization

This guide consists of 13 chapters and four appendixes:

Chapter 1	Defines the tasks that make up the job of a Digital UNIX system administrator and points to sources of information about these tasks in this manual and other places.
Chapter 2	Describes methods and tools for system administration tasks.
Chapter 3	Explains how to start up and shut down the operating system. Additionally, explains how to recover from an unexpected shutdown.
Chapter 4	Describes how to customize certain operating system files and diverse operating system components in order to tailor the operating system environment.
Chapter 5	Describes how to dynamically and statically configure an operating system kernel.

Chapter 6	Describes how to administer the SCSI Dynamic Device Recognition capabilities of the operating system. Additionally, it explains how to administer the terminals and other mass storage devices that are configured into the operating system.
Chapter 7	Explains how to administer the UNIX file system (UFS).
Chapter 8	Explains how to administer the POLYCENTER Advanced File System (AdvFS).
Chapter 9	Explains how to administer the Logical Storage Manager (LSM).
Chapter 10	Explains how to administer accounts for operating system users and groups of users.
Chapter 11	Explains how to administer the print services of the operating system.
Chapter 12	Explains how to administer the archiving services of the operating system in order to backup and restore mass storage devices.
Chapter 13	Explains how to administer the resource accounting services of the operating system.
Chapter 14	Explains how to prevent errors by using system exercisers and how to set up and administer the error logging services of the operating system.
Appendix A	Contains information about device mnemonics.
Appendix B	Contains information about the SCSI/CAM Utility Program.
Appendix C	Contains information about the CI bus and the Hierarchical Storage Controller (HSC) configuration.
Appendix D	Contains information about the `uerf` event logger, a component that will be retired in a future version of the operating system.
Appendix E	Contains information about specific hardware devices that are supported in this release. Instructions for installing and configuring the devices is also provided

Related Documents

The *Installation Guide* describes how to install your operating system. The *Network Administration* manual describes how to set up, configure, and troubleshoot your network.

The printed version of the Digital UNIX documentation set is color coded to help specific audiences quickly find the books that meet their needs. (You can order the printed documentation from Digital.) This color coding is reinforced with the use of an icon on the spines of books. The following list describes this convention:

Audience	Icon	Color Code
General users	G	Blue
System and network administrators	S	Red
Programmers	P	Purple
Device driver writers	D	Orange
Reference page users	R	Green

Some books in the documentation set help meet the needs of several audiences. For example, the information in some system books is also used by programmers. Keep this in mind when searching for information on specific topics.

The *Documentation Overview, Glossary, and Master Index* provides information on all of the books in the Digital UNIX documentation set.

Reader's Comments

Digital welcomes any comments and suggestions you have on this and other Digital UNIX manuals.

You can send your comments in the following ways:

- Fax: 603-881-0120 Attn: UEG Publications, ZK03-3/Y32
- Internet electronic mail: `readers_comment@zk3.dec.com`

  A Reader's Comment form is located on your system in the following location:

  `/usr/doc/readers_comment.txt`

- Mail:

  Digital Equipment Corporation
  UEG Publications Manager
  ZK03-3/Y32
  110 Spit Brook Road
  Nashua, NH 03062-9987

  A Reader's Comment form is located in the back of each printed manual. The form is postage paid if you mail it in the United States.

Please include the following information along with your comments:

- The full title of the book and the order number. (The order number is printed on the title page of this book and on its back cover.)
- The section numbers and page numbers of the information on which you are commenting.
- The version of Digital UNIX that you are using.
- If known, the type of processor that is running the Digital UNIX software.

The Digital UNIX Publications group cannot respond to system problems or technical support inquiries. Please address technical questions to your local system vendor or to the appropriate Digital technical support office. Information provided with the software media explains how to send problem reports to Digital.

Conventions

This guide uses the following conventions:

% $	A percent sign represents the C shell system prompt. A dollar sign represents the system prompt for the Bourne and Korn shells.
#	A number sign represents the superuser prompt.
file	Italic (slanted) type indicates variable values, placeholders, and function argument names.
[|] { | }	In syntax definitions, brackets indicate items that are optional and braces indicate items that are required. Vertical bars separating items inside brackets or braces indicate that you choose one item from among those listed.
. . .	In syntax definitions, a horizontal ellipsis indicates that the preceding item can be repeated one or more times.
⋮	A vertical ellipsis indicates that a portion of an example that would normally be present is not shown.

`cat`(1)	A cross-reference to a reference page includes the appropriate section number in parentheses. For example, `cat`(1) indicates that you can find information on the `cat` command in Section 1 of the reference pages.
Ctrl/*x*	This symbol indicates that you hold down the first named key while pressing the key or mouse button that follows the slash. In examples, this key combination is enclosed in a box (for example, Ctrl/C).
Return	In an example, a key name enclosed in a box indicates that you press that key.

1

Overview of Digital UNIX System Administration

This chapter surveys many of the tasks that are performed by Digital UNIX system administrators and points to the places in this manual and in other sources that describe these tasks.

1.1 The Digital UNIX System Administrator

As administrator of a Digital UNIX operating system, you work in the following envrionment:

- The hardware and software

  The hardware environment is a client Alpha workstation in a local area network being served by an Alpha server. The client and server machines are running supported versions of the Digital UNIX operating system software.

- The administrative tasks

  As administrator, you perform operational tasks to set up, monitor, maintain, and exploit available software and hardware resources.

- The user interface to the operating system

  For many system administration tasks, you can choose to use the UNIX command line interface or the *SysMan* graphical user interface.

  This manual describes the UNIX command line interface for the tasks.

  The *SysMan* interface is now recommended by Digital for interacting with the operating system. If possible, you should examine whether or not you can use this interface for your administrative tasks. See Section 2.2 for more information.

This manual does not include information about planning an operating system environment. For information about planning operating system environments, see your local Digital representative.

Information about administering network-related tasks is documented in the Digital UNIX *Network Administration* manual.

The information you need to install the Digital UNIX operating system is contained in the Digital UNIX *Installation* guide, and all the information

you need to update the Digital UNIX operating system is contained in the Digital UNIX *Update Installation* card.

The following sections survey the contents of the chapters in this manual. In general, the first few chapters contain information about the setup tasks you perform the first few hours after installing or updating new operating system software.

The several remaining chapters contain information about the schedule- and event-driven tasks that make up the bulk of a system administrator's work. These tasks use the operating system components, its services, or features to maintain the performance of the operating system and the satisfaction of its users.

1.2 Starting Up and Shutting Down the System

Chapter 3 contains instructions for booting and for shutting down systems. You will need this information for the following tasks:

- System testing or troubleshooting

  When you test or troubleshoot your system, you need to move between run levels, for example, from multiuser mode to single-user mode.

- Setting the system clock

  If your system has been powered down for an extended amount of time, you will need to set the system clock.

1.3 Customizing the System Environment

Periodically, you will need to tailor your system to fit your needs. For example, you might need to:

- Modify system files to add information about new devices or to invoke certain programs when the system moves from one run level to another.
- Specify commands to be run automatically at a specific time.
- Set your system to a different locale. The Digital UNIX operating system includes National Language Support (NLS) for multiple environments.

Chapter 4 describes how to customize your system for these purposes, and also discusses how to customize:

- Internationalization features for programmers and users
- Your time zone
- The environment needed to establish a system that meets the requirements of your site's security policy

- The tools you use to maintain adequate performance for operating system components and for user applications
- The power consumption of certain hardware subsystems

1.4 Configuring the Kernel

You may need to tune your system to enhance performance, add new devices, or install new software. These changes may require you to modify your system configuration file. If you modify the system configuration file, thereby reconfiguring the kernel, you must compile and boot a new system image for the changes to take effect. Chapter 5 includes the following information:

- Descriptions of the configuration files and the tunable options in the files
- Instructions for configuring, compiling, and booting a new kernel
- Instructions for configuring a STREAMS module or driver into your system
- Instructions for administering loadable drivers

1.5 Administering Dynamic Device Recognition

Chapter 6 describes the components you use and the tasks you perform in order to administer the Dynamic Device Recognition capabilities of your operating system. Dynamic Device Recognition (DDR) is a framework for describing the operating parameters and characteristics of SCSI devices to the SCSI CAM I/O subsystem.

You use the `ddr_config`(8) utility and the `ddr.dbase`(4) text database to make changes to the subsystem whenever you change the devices in the SCSI CAM I/O subsystem. You make these changes after the operating system is installed and without needing to reboot the operating system.

The administrative tasks are:

- Compiling the DDR database
- Converting the `cam_data.c` file to entries in the DDR database

You use the `ddr_config` utility to accomplish both tasks.

In addition, this chapter contains instructions for maintaining other terminals and mass storage devices on your system. The tasks involved include the following:

- Making a new kernel (for disk and tape devices only)
- Modifying the proper system files

- Making devices known to your system

1.6 Administering the UNIX File System

Chapter 7 contains information to help you perform the following file system and disk management tasks:

- Allocate more swap space or change your method of swap space allocation
- Create, check, tune, maintain, and mount file systems
- Determine how your disk space is being utilized
- Repartition the file systems on a disk

1.7 Administering the Advanced File System

Chapter 8 describes the Advanced File System (AdvFS) component. AdvFS offers rapid crash recovery, high performance, and a flexible structure that enables you to manage your file system while it is on line.

1.8 Administering the Logical Storage Manager

The Logical Storage Manager (LSM) software has disk management capabilities that increase data availability and improve disk I/O performance. System administrators use LSM to perform disk management functions dynamically without disrupting users or applications accessing data on those disks. Chapter 9 describes the elements of LSM that are most commonly used by a system administrator.

1.9 Administering User Accounts and Groups

Use the information in Chapter 10 to perform the following tasks:

- Add, modify, and remove user accounts
- Add and remove groups

1.10 Administering the Print Services

Use the information in Chapter 11 to perform the following printer management tasks:

- Add and remove printers and change the configuration of an existing printer
- Show the status of a printer and control the printer; for example, delete print requests and enable and disable printers

- Enable printer accounting

1.11 Administering the Archiving Services

Use the information in Chapter 12 to perform the following tasks:

- Choose which file systems to back up so that all the files on your system, data files as well as system files, are protected from loss
- Choose a backup schedule so that you do not have to back up all the files in a file system at each backup
- Set up a schedule for performing a full backup of each file system on your entire system, including all the system software
- Set up a routine backup schedule to make it easier to remember which backup to do each day
- Use the `rdump` command to make backups on a remotely located tape device
- Automate the backup process by using shell scripts
- Restore a file system by using the `restore` or the `vrestore` command
- Restore files from a remote tape device by using the `rrestore` command
- Restore the root and `/usr` file systems

1.12 Administering System Accounting Services

Chapter 13 describes how to set up and use accounting to track system resources. It describes how to perform the following accounting administration tasks:

- Set up the resource accounting software
- Enable automatic accounting
- Create accounting reports
- Make sure that the accounting files are a manageable size

1.13 Administering Events and Errors

Chapter 14 contains information on system events and errors. First, it describes how you can use the system exercisers to discover potential system problems. With this information you can act to prevent events or errors from occurring. Next, the chapter discusses system events and the components you employ to react to these events. Use the information in this chapter to perform the following tasks:

- Change the default event-logging configuration
- Examine the event-logging files to ensure that problems do not exist
- Make sure that the event-logging files are a manageable size

1.14 Appendixes

The appendixes to this manual describe auxiliary information, seldom-used utilities, and components which are scheduled for retirement, but that are documented for backward compatibility.

1.14.1 Device Mnemonics

Appendix A identifies and defines the mnemonics that you use to attach any hardware or software device to your system. You specify the mnemonics when you create the character or block special files that represent each of the devices.

1.14.2 SCSI/CAM Utility Program

Appendix B describes the SCSI/CAM Utility Program (SCU), which interfaces with the Common Access Method (CAM) I/O subsystem and the peripheral devices attached to Small Computer System Inferface (SCSI) busses. This utility implements the SCSI commands necessary for manual maintenance and diagnosis of SCSI peripheral devices and the CAM I/O subsystem.

1.14.3 CI and HSC Hardware

Appendix C describes the CI bus and Hierarchical Storage Controller (HSC) hardware. This appendix contains information about hardware and software installation, setup, and restrictions. It also describes how to boot an HSC controller and disk and how to share HSCs among hosts.

1.14.4 Using the uerf Error Logger

The `uerf` error logging utility has been scheduled for retirement. Its use is described here for backward compatibility purposes.

1.14.5 Administering Specific Hardware Devices

Appendix E Explains how to install specific hardware devices available on some processors.

2

System Administration Tools and Methods

2.1 Scripts and Files

Most of the tasks described in this book can be accomplished by using a combination of individual shell commands, using scripts and utilities, and by editing system files. This chapter describes:

- Administration tools available from the Common Desktop Environment (CDE) graphical user interface. Use of this interface requires a graphics (windowing) terminal or workstation, and the installation of the CDE software subsets.
- Remote system administration, using a modem connected to the console of the remote system

2.2 CDE Graphical User Interface

CDE is the the preferred operating system interface and the *SysMan* graphical user interface becomes the preferred system administration tool on systems that can display the CDE. A command line interface to these tools is included for users of systems that have only character-cell displays or for users who prefer to use a command line interface to the CDE in a terminal window.

Documentation on using the tools is provided in the form of on-line help, and context-sensitive messages. The *SysMan* interfaces are not documented in this book However, you will find information relating to administrative tasks, and instructions for using alternative methods.

2.2.1 CDE Administration Tools

The following *SysMan* tools are available from the CDE control panel:

- Installation Applications

  You can use these applications to set up installations, preconfigure some software subsystems, and access the UNIX shell.
- Configuration Applications

After a system has been installed, you can use these applications to perform:

- Network Configuration with `netconfig`
- BIND Configuration with `bindconfig`
- NFS Configuration with `nfsconfig`
- Mail Configuration with `mailconfig`
- Print Configuration with `printconfig`
- Disk Configuration with `diskconfig`

- Daily Administration Applications

  After a system has been configured, you can use these applications to perform routine administrative tasks:

  - Account Manager with `dxaccounts`
  - Archiver with `dxarchiver`
  - File Share with `dxfileshare`
  - Host Manager with `dxhosts`
  - License Manager with `dxlicenses`
  - Shutdown Manager with `dxshutdown`
  - System Information with `dxsysinfo`

- Monitoring and Tuning Applications

  While a system is running, you can use these applications to monitor and tune its resources:

  - Kernel Tuner with `dxkerneltuner`
  - Process Tuner with `dxproctuner`

- Display Window Help Application

  You can use the `dxdw` application to access the CDE commands from the graphical user interface.

2.2.2 Accessing the SysMan Tools

The SysMan applications are also available in the DECwindows Motif and base X Windows graphical environments. In CDE, the SysMan applications are available in the `Application Manager`. You can access the `Application Manager` from the CDE Front Panel by clicking on its icon. The SysMan applications are organized into five groups within the System_Admin group. Double click on the `System_Admin` group to access the `SysMan Configuration Checklist`, the `Welcome to SysMan` online help volume, and the five application groups.

Online help is available for the SysMan applications without running the applications. Click on the Help Manager icon on the CDE front panel to display the online help browser. The browser includes help families for CDE, the CDE Desktop, and Digital System Management.

In DECwindows, the SysMan applications are listed in the Session Manager's Options menu. You can use the Applications Definitions menu item to add frequently used SysMan applications to the Applications menu. You can also customize your CDE workspace with the Create Action utility in the Desktop_Apps folder. Customized icons enable you to start SysMan applications directly from the workspace.

In other X Windows environments, the SysMan applications can be invoked from the command line. See the sysman_intro(8X) reference page for a list of the SysMan applications. This reference page also describes how to display the online help browser in graphical environments other than CDE.

To support nongraphical (terminal) environments, some SysMan applications offer command line and question and answer interfaces. The following applications have a command line interface. A single command starts the application, which then performs the actions specified by the command-line arguments.

- Network Configuration
- BIND Configuration
- NFS Configuration
- Mail Configuration
- Account Manager

The following applications have a question and answer interface invoked using the command-line argument **-ui menu**. The application prompts you interactively.

- Network Configuration
- BIND Configuration
- NFS Configuration
- Printer Configuration

The menu interface for Mail Configuration is called mailsetup.

2.3 Remote System Administration

You can manage remote systems through a modem connection. A serial line console enables you to connect a local terminal to the remote system console through modems attached to your local system and to the

communications port COMM1 of the remote system. The local system can be any terminal or terminal emulation device that enables a modem connection such as a dumb terminal, an X terminal window, or a personal computer (PC). To perform administrative tasks, you must be able to log in as root (or an account with administration privileges).

This connection is referred to as the **console port**. The terminal connection supports a limited set of communication rates up to 57,600, depending on the console firmware supported by your processor. Currently, this feature is only available on systems that support modems as console devices, such as the AlphaServer 1000A. Consult your system hardware documentation to find out if your system has such capabilities.

The console port enables you to do the following:

- Connect to a remote system using a utility such as `tip`, `telnet`, or a PC terminal emulation utility
- Remotely boot or shut down a system and observe all the boot messages
- Start the kernel debugger and observe debugging messages
- Perform any system administration tasks using commands and utilities

Note that running the Environment Configuration Utility (ECU) on the remote system will cause the modem to disconnect. For this reason, you should use the ECU to complete any environment configuration before setting up and using a modem as a console device.

2.3.1 Setting Up a Console Port

The following sections provide an overview of the steps required to set up a serial line console port and set up the remote modem for dial-in. It is assumed that your local (dial-out) modem is already installed and configured for use.

2.3.1.1 Connecting the Modem to COMM1

The `CONSOLE` environment variable on the remote system should be set to `serial`.

Refer the hardware documents supplied with your modem for connecting the modem to your system. Consult the `modem`(7) reference page to obtain the correct modem settings and for instructions on how to create the appropriate system file entries. In particular, the `cons` entry in `/etc/inittab` file should be modified so that the `getty` or `uugetty` process sets up the COMM port correctly. This line is similar to the following example:

```
cons:1234:respawn:/usr/sbin/getty console console vt100
```

This line should be changes as follows if you are using a modem set to run at a baud rate of 38,400 as a console device:

```
cons:1234:respawn:/usr/sbin/getty console M38400 vt100
```

2.3.1.2 Setting the Configurable DCD Timer Value

The Digital UNIX serial driver has been modified to allow the Carrier Detect (DCD) timeout value to be configurable. The default value for this timer is 2 seconds, which is in accordance with the DEC STD-052 standard and is acceptable for most modems. This timer is used to determine how long the driver must wait when the DCD signal drops, before declaring the line disconnected and dropping the DTR and RTS signals. Some modems expect DTR to drop in a shorter time interval, so refer to your modem documentation to verify the interval.

The timer can be modified via the `/etc/sysconfigtab` file or the `sysconfig` command to set the timer to 0 (no timeout period), 1, or 2 seconds. To set the timer via `/etc/sysconfigtab`, edit the file and include the following:

```
ace:
  dcd_timer=n
```

Where n = 0, 1, or 2

The syntax for modifying the timer via the sysconfig command is as follows:

```
# sysconfig -r ace dcd_timer=n
```

Where n = 0, 1, or 2

Note that by modifying the value with the `sysconfig` command, the setting is lost when the system is rebooted. To preserve the setting across reboots, edit the `/etc/sysconfigtab` file.

2.3.1.3 Setting the Console Environment Variables

The `COM1_MODEM`, `COM1_FLOW`, and `COM1_BAUD` console environment variable settings must be equivalent to the `getty` or `uugetty` settings used when you created your system file entries for the modem.

Consult your hardware documentation for information on how to set the console environment variables. Typically, the variables are set when the system is shut down and in console mode, as shown in the following example:

```
>>> set COM1_MODEM ON
>>> set COM1_FLOW SOFTWARE
```

```
>>> set COM1_BAUD 9600
```

Valid settings are as follows:

- COM1_MODEM: ON or OFF
- COM1_FLOW: NONE, HARDWARE, SOFTWARE, BOTH
- COM1_BAUD: Consult your system hardware documentation.

Note that if you change the baud rate, flow control, or modem setting (for example, using the getty command), the change will be propagated down to the console level and the environment variables will change automatically.

2.3.1.4 Verifying the Modem Setup

Dial the remote system and obtain a log-in prompt or console prompt, if the system is not booted. Log out or disconnect and ensure that the line hangs up correctly. Dial in again to ensure that you can reconnect.

2.3.2 Initiating a Console Port Connection

You can initiate a connection between the local and remote systems by different methods. A tip, kermit, or cu connection can be initiated from a terminal or X-terminal window or you can use a PC-based terminal emulator.

For example, use the tip command as follows:

```
# tip [telephone number]
# tip cons
```

Where telephone_number is the telephone number of the remote system, including any prefixes for outside lines and long-distance codes. The second line is an example of an entry in the /etc/remote file, which you can use to specify details of remote systems and tip settings.

Once you have initiated the dial-out command, and the two modems have established a connection, the word connect is displayed on your local terminal window. Press the Return key and the console prompt (>>>) or the login: prompt will be displayed.

See the tip(1) reference page for more information.

2.3.2.1 Using the Console Port

Once you have access to the system and are logged in to a privileged account, you can perform any of the administration tasks described in this

volume that do not require access to a graphical user interface, such as using commands and running utilities. Note that the following Digital UNIX features may be useful for remote administration:

- The UNIX-to-UNIX system copy utility, `uucp` for copying scripts and files to the remote system. See the `uucp`(1) reference page for more information.
- A kernel debugging tool, `ikdebug` can be invoked and used remotely. See the `ikdebug`((8)) reference page for more information. You may need to change an entry in the `/etc/remote` file to correct the baud rate. For example you may need to change the baud rate from 9600 baud in the following lines:

  ```
  # access line for kernel debugger
  kdebug:dv=/dev/tty00:br#9600:pa=none:
  ```

  See the *Kernel Debugging* manual for additional information.

2.3.2.1.1 Turning off Console Log Messages

The `syslogd` daemon now has an internal switch to disable and enable messages to the console. This feature is invoked by the `-s` flag on the `syslogd` command line, or by running the following command:

```
# /usr/sbin/syslog
```

See the `syslog`(1) reference page.

2.3.2.1.2 Shutting Down The Remote System

When you shut down the remote system, the modem connection will be dropped. To avoid this, use the following command before you shut down the system:

```
# stty -hupcl
```

See the `stty`(1) reference page for more information.

When the shutdown is complete, you will still have access to the console prompt.

2.3.2.1.3 Ending a Remote Session

To end a remote session from the Digital UNIX operating system shell prompt, type Ctrl-D to log out and terminate the remote session. Otherwise, type `+++` to put the modem into local command level, and type `ATH` followed by the Return key to hang up the connection.

2.3.3 Troubleshooting

If you have problems setting up your systems and connecting, check the set up as follows::

- The local modem does not dial out.

  Check the cables and connections and ensure that the telephone lines are plugged into the correct sockets, and that you have a dial tone.

- The remote modem fails to answer.

  Ensure that the remote modem is set to auto-answer, `ATS0=n`, where `n` is the number of rings before the modem answers.

  Review the `modem`(7) reference page and verify the settings for dial-in access.

- The remote modem answers and then disconnects.

  This is most likely to be due to incorrect settings for dial-in access. Review the `modem`(7) reference page and verify the settings for dial-in access.

- The remote modem answers but only random characters are printed.

  This problem is usually caused by a mismatch between the baud rate of the COMM port and that of the modem. Review the `modem`(7) reference page and verify the settings for dial-in access.

- The connection is dropped when the remote system is shut down via the `shutdown` command.

  The `stty` attribute `hupcl` is at the default setting. To prevent the line from disconnecting during a shut down, use the following command:

  ```
  # stty -hupcl
  ```

3

Starting Up and Shutting Down the System

This chapter describes procedures for starting up and shutting down the operating system and includes a discussion of:

- Boot operation
- Different startup states and the corresponding boot preparation
- Run levels
- Resolving problems that occur during the boot operation

Refer to the *Installation Guide* for information about installing the system and performing the initial boot operation. The information in this chapter assumes that you are booting or rebooting an installed operating system.

Shutting down the system is a routine task that you should perform periodically. Usually, you can shut down the system easily and with minimal disruption to system users. Occasionally, you must shut down the system rapidly, causing a moderate degree of disruption to users. Under some circumstances (that are out of your control), the system shuts itself down suddenly, causing substantial disruption to users.

3.1 Understanding the Boot Operation

When you boot the operating system, you initiate a set of tasks that the system must perform to operate successfully. The system is vulnerable during startup since it is loading the kernel into memory and initializing routines that it depends on for operation. Consequently, you should understand what is happening during the system boot, and be prepared to respond if problems occur.

Although certain boot operations are hardware dependent, some features typically apply to all systems. For example:

- The system always boots either automatically or manually.

  In an automatic boot, the system controls the entire operation. When you boot the system to multiuser mode, or shut down the system with the reboot flag, or when the system panics and recovers, you are relying

on an automatic boot. With an automatic boot, the system begins the initialization process and continues until completion or failure.

Manual intervention may be required if the automatic boot fails for some reason, for example, if the `fsck` command fails.

In a manual boot, the system controls the initial operation, turns control of the procedure over to you, then reinstates control to complete the operation. When you boot the system to single-user mode, you are relying on a manual boot. In an automatic or a manual boot, the operation either succeeds or fails:

- If the boot operation succeeds, the system is initialized. In single-user mode, the system displays the root prompt (#) on the console or on the terminal screen. In multiuser mode, the system displays the login prompt or a startup display. The prompt or startup display differs according to hardware capability and available startup software.
- If the boot operation fails, the system displays an error message followed by a console firmware prompt (>>>). In the worst case, the system hangs without displaying a console prompt.

- The system boots to either single-user or multiuser mode.
  - In a boot to single-user mode, the software loads the kernel and proceeds through the initialization tasks associated with process 0 (initialization) and process 1 (init). The `init` program creates a Bourne shell (`sh`), turns control over to you, and waits for you to exit from the shell with the `exit` command or Ctrl/d before continuing with its startup tasks.

    Because `init` does not invoke the startup script prior to turning control over to you, the root file system is mounted read only, startup of the network and other daemons does not occur, file checking and correction are not enabled, and other operations necessary for full system use are not automatically available to you.

    Usually you boot to single-user mode to perform specific administrative tasks that are best accomplished without the threat of parallel activity by other users. You perform these tasks manually before exiting from the Bourne shell. For example, you might check new hardware, mount and check aberrant file systems, change disk partitions, or set the system clock. When you finish your work, you return control to the system, and `init` continues with its startup tasks and boots to multiuser mode.
  - In a boot to multiuser mode, the system loads the kernel and moves through various phases such as hardware and virtual memory initialization, resource allocation, scheduling, configuration, module

loading, and so on. At the conclusion of the main initialization tasks (process 0), `init` (process 1) starts an additional set of tasks that includes reading the `/etc/inittab` file, acting on instructions found there, and executing the relevant run command scripts. These scripts contain entries that initiate activities such as mounting and checking file systems, removing temporary files, initializing the clock daemon, initializing the network daemon, setting up printer spooling directories and daemons, enabling error logging, and performing other tasks specified within the scripts or in related directories. At the conclusion of these activities, the system is enabled and accessible to users.

The Digital UNIX operating system allows you to boot an alternate kernel. For example, if you cannot boot your system, you could boot `/genvmunix` to troubleshoot the problem with your system. You could also boot an alternate kernel to test new drivers or to add options to the existing kernel.

3.2 Preparing to Boot the Installed System

As the system administrator, you set up or encounter various preboot or postshutdown states. This section describes and recommends procedures for preparing and initiating a reboot from a variety of system states. The states discussed here include the following:

- A powered-down system
- A powered-up, halted system
- A powered-up system in single-user mode
- A crashed system

Note

If the system is running in single-user mode and you want to use the `ed` editor, you must change the protections of the root file system to read-write. At the prompt, enter the following command:

```
# mount -u /
```

3.2.1 Preparing to Boot a Powered-Down System

A system is powered down when the hardware (processor, devices, and peripherals) is turned off. Administrators power down the hardware periodically for routine maintenance or to configure new devices.

If you are preparing to reboot your system from a powered-down state, follow these steps:

1. Confirm that the hardware and all peripheral devices are connected. Refer to the operator's guide for your hardware for information and instructions for interpreting diagnostic output.
2. Power up the hardware and peripheral devices. Remember to power up all devices that you powered down earlier. Refer to the operator's manual or the hardware user's guide for instructions on starting your hardware and peripherals.
3. Confirm that the hardware completed its restart and diagnostic operations. Most hardware provides a diagnostic check as a routine part of its startup operation. Refer to the operator's manual for your hardware for information about your hardware's restart and diagnostic operations.
4. Wait for the console prompt (>>>). If you have enabled your system to boot automatically upon powerup, press the halt button to display the console prompt. Refer to the hardware operator's guide for the location of the halt button on your system. See Section 3.3 for more information on setting the default boot action for your system.
5. Decide which startup mode you want to initiate:
   - If you have tasks you need to accomplish and want the system to restrict access to all users but root, plan to boot to single-user mode.
   - If you do not require single-user access and you want the system to initialize full functionality, plan to boot to one of the multiuser modes: multiuser without networking or multiuser with networking.
6. Enter the boot command that corresponds to the desired startup mode.

   Refer to Section 3.3 for the commands and procedures required to boot your system.

3.2.2 Preparing to Boot a Powered-Up, Halted System

When your machine is powered up and enabled but the processor is halted, the system is in console mode. For example, after you shut down the processor with the `shutdown -h` command or when you run the `halt` command, your system displays the console prompt (>>>).

When the system displays the console prompt, follow these steps to prepare to boot your system:

1. Decide which startup mode you want to initiate:

- If you have tasks you need to accomplish and you want the system to restrict access to all users but root, plan to boot to single-user mode.
- If you do not require single-user access and you want the system to initialize full functionality, plan to boot to one of the multiuser modes: multiuser without networking or multiuser with networking.

2. Enter the boot command that corresponds to the desired startup mode.

   Refer to Section 3.3 for the commands and procedures required to boot your system.

3.2.3 Preparing to Transition from Single-User Mode

When your machine is powered up and enabled, the processor is running, and access is limited to root, the system is in single-user mode.

When the system displays the single-user prompt (#), follow these steps to prepare to go to multiuser mode:

1. Decide if you should continue in single-user mode or if you should go to multiuser mode:
   - If you have additional tasks that you need to perform and you want the system to deny access to all users but root, plan to continue in single-user mode.
   - If you do not require single-user access, or if you have completed your tasks and you want the system to initialize full functionality, plan to go to one of the multiuser modes: multiuser without networking or multiuser with networking.
2. When you are ready to go to multiuser mode, press Ctrl/d. Refer to Section 3.3 for the commands and procedures required to boot your system.

3.2.4 Preparing to Boot a Crashed System

If your system crashes and is unable to recover automatically and reboot itself, follow these steps to prepare to boot the system:

1. Confirm that the hardware and all peripheral devices are connected.
2. Power up the hardware, if necessary. Always power up peripherals and devices before the processor.
3. Monitor the hardware restart and diagnostic operations. Refer to the operator's guide for your hardware for information and instructions for interpreting diagnostic output.

- In the unlikely event that the diagnostic test indicates hardware failure, contact your Digital field representative. Because hardware damage is a serious problem, do not continue or try to bypass the defective hardware.
- If you have enabled your system to boot automatically, press the halt button to display the console prompt. Refer to the hardware operator's guide for the location of the halt button on your system.

4. Decide which startup mode you want to initiate:
   - If you need to deny access to all users but root, plan to work in single-user mode. After a crash, it is wise to work initially in single-user mode. You should check all file systems thoroughly for inconsistencies and perform other postcrash operations before enabling system access to other users.
   - If you need to allow access to you and to all other users with login permission, plan to boot to one of the multiuser modes: multiuser without networking or multiuser with networking.
5. Enter the required boot command.

   Refer to Section 3.3 for the commands and procedures required to boot your system.

3.3 Booting the System

The command that you use to boot the kernel depends on several factors:

- Processor type
- Run level
- Location of the kernel that you are booting (on the system disk or on a remote server)
- Whether you are booting all processors or a single processor (in a multiprocessor system)
- Whether any console environment variables are defined
- Whether you are booting the default kernel or an alternate kernel

3.3.1 Defining the Console Environment Variables and Using the Boot Commands

To boot your system you need to understand the use of certain console environment variables and their role in affecting the boot process. Table 3–1 lists each of the console environment variables and their associated actions. If you are booting a DEC 2000 processor, refer to the hardware manual that accompanied the processor for information on boot commands.

Table 3–1: Console Environment Variables

Variable	Action
boot_reset	When set to on, resets the hardware on boot
boot_osflags	A combination of flags used to control the boot loader and kernel
bootdef_dev	Identifies the boot device
boot_file	Identifies the kernel to boot (on DEC 4000 and DEC 7000 processors)
cpu_enable	Selectively enables particular processors from the console

To prepare the hardware:

1. Set the auto_action variable to halt:

   ```
   >>> set auto_action halt
   ```

   The previous command will halt the system at the console prompt each time your system is turned on, when the system crashes, or when you press the halt button.

2. For DEC 3000 and DEC 7000 processors, set the boot_reset variable to on to force the resetting of the hardware before booting:

   ```
   >>> set boot_reset on
   ```

3. For DEC 3000 processors, set the time to wait to reset the SCSI device before booting:

   ```
   >>> set scsi_reset 4
   ```

4. Use the following procedure to set the boot_osflags variable and the boot device:

   a. Determine which options to the boot_osflags variable you want. Table 3–2 lists the options.

Table 3–2: Options to the boot_osflags Variable

Option	Action
a	Boot to multiuser mode. (By default, the kernel boots to single-user mode.)
k	Use the kdebug debugger to debug the kernel. (By default, kdebug is not used.)
d	Use full crash dumps. (By default, partial dumps are used.)
i	Prompt for the kernel and special arguments. (By default, no questions are asked.)

The options are concatenated into the boot_osflags variable to achieve the desired effect. For example, to boot to multiuser mode and use full crash dumps, enter:

```
>>> set boot_osflags ad
```

If you want the defaults, clear the variable as shown in the following example:

```
>>> set boot_osflags ""
```

b. Determine the unit numbers for your system's devices:

```
>>> show device
```

If you want to boot from the dual SCSI TURBOchannel option card (PMAZB or PMAZC), complete the following steps:

i. Identify the slot number of the PMAZB or PMAZC option card:

```
>>> show conf
```

The previous command displays the system configuration.

ii. Determine the unit number of your system's devices:

Use the conf command with the slot number to identify the unit numbers of the devices attached to that controller. For example, to look at the devices attached to the controller in slot 1, enter:

```
>>> t tc1 cnfg
```

A display appears identifying the unit number of each device attached to that controller. Identify the unit number of the device from which you want to boot.

c. Set the default boot device:

By default, you must provide a boot device when you boot your system. If you always boot from the same device, use the following command syntax with the bootdef_dev variable to set a default boot device:

set bootdef_dev *device*

For example, to boot the system off of disk dka0, enter:

```
>>> set bootdef_dev dka000
```

To boot the system from the first disk on the PMAZB or PMAZC option card in TURBOchannel slot 1, enter the following command. Note that the double quotes (") are necessary for the console to understand where it is booting from.

```
>>> set bootdef_dev "1/dka000"
```

d. You have the option of booting from an alternate kernel. If you want to do this, enter:

```
>>> set boot_osflags i
```

When booting, the system prompts you to enter a file name. For example:

```
Enter [kernel_name] [option_1 ... option_n]: genvmunix
```

The system will display informational messages.

On DEC 4000 and DEC 7000 processors, you can also boot an alternate kernel by setting the `boot_file` variable to the name of the kernel you want to boot. For example, to boot a `genvmunix` kernel, enter:

```
>>> set boot_file genvmunix
```

On DEC 4000 and DEC 7000 processors, you must clear the `boot_file` variable if you want to boot the default kernel, `/vmunix`. For example:

```
>>> set boot_file ""
```

In a multiprocessor configuration, you can use the `set cpu_enable` command to selectively enable processors from the console. The mask is a bit field, where each bit represents a slot position. The easiest way to ensure all processors are enabled is to set the CPU mask to `ff`. After setting the mask, turn the power on the system off and then back on again.

The operating system also provides a mechanism for enabling or disabling processors at system boot time. See the description of the `cpu-enable-mask` attribute in the *System Tuning and Performance Management* guide for information.

After you have set the console variables, use the following command to boot the system:

```
>>> b
```

3.3.2 Overriding the Boot Commands

The previous section described how to set the boot commands. This section describes how to override those commands.

- Overriding `bootdef_dev`

  To override the `bootdef_dev` variable, supply the desired boot device as an argument to the boot command. For example, if your boot device is set to boot from disk `dka0` and you want to boot from disk `dkb0`, enter:

```
>>> b dkb0
```

- Overriding boot_osflags

  The boot_osflags variables are ignored if you specify the –fl option to the boot command, as follows:

  ```
  >>> b -fl
  ```

  To override the boot_osflags variables, pass the desired choices to the –fl option. For example, the following command boots to the interactive prompt so you can specify an alternate kernel, and then boots to multiuser mode:

  ```
  >>> b -fl ai
  ```

- Overriding boot_file

  To boot a kernel other than that specified by boot_file, enter the boot command with the following syntax:

  b -fi *kernel*

  For example, to boot the genvmunix kernel, enter:

  ```
  >>> b -fi genvmunix
  ```

3.4 Identifying the System Run Levels

A run level (mode) specifies the state of the system and defines which processes are allowed to run at that state. There are four basic run levels available, as follows:

0	Specifies the halt state
S or s	Specifies single-user mode
2	Specifies multiuser mode without network services
3	Specifies multiuser mode with network services
	Specifies the console mode

The inittab file contains line entries that define the specific run levels and the run command scripts that are associated with the run level. When the init process starts, it reads the inittab file and executes the relevant run command scripts. The scripts, in turn, define which processes are to run (and which processes are to be killed if the system is changing from one level to another) at a specific run level. Refer to the init(8) and inittab(4) reference pages and to Chapter 4 for information about reading and modifying the inittab file.

3.5 Changing the System Run Levels

Before changing to a new run level, check the `inittab` file to confirm that the run level to which you intend to change supports the processes you need. Of particular importance is the `getty` process since it controls the terminal line access for the system console and other logins. Make sure that the `getty` entry in the `inittab` file allows system console access at all run levels. Refer to the `inittab`(4) reference page for more information about defining run levels. Refer to the `getty`(8) reference page for more information about defining terminal lines and access.

Before changing to a new run level, use the `wall` or `write` command to warn users that you intend to change the run level. Since a change in run level could result in termination of the user's `getty` process (which disables their login capability) as well as termination of other processes that they are running, you should communicate the change to each logged in user. Check the `getty` entry for user terminals to verify that the new run level is specified in the entry. If it is not, request that users log off so that their processes will not terminate in response to a `kill` signal from `init`.

When the system is initialized for the first time, it enters the default run level that is defined by the `initdefault` line entry in the `inittab` file. The system continues at that run level until `init` receives a signal to change run levels. The following sections describe these signals and provide instructions for changing run levels.

3.5.1 Changing Run Levels from Single-User Mode

Use the Bourne shell when working in single-user mode and press Ctrl/d to change run levels. The shell terminates in response to Ctrl/d and displays the following message if transitioning from single-user mode to multiuser mode during a boot operation:

```
INIT: New run level: 3
```

If this transition is made from single-user mode with the previous state having been multiuser mode, then a prompt is issued for input of the desired run level. The `init` process searches the `inittab` file for entries (at the new run level) with the `boot` or `bootwait` keywords, and then acts on these entries before it continues with the normal processing of the `inittab` file. The `init` process next scans the file for other entries with processes that are allowed to run at the new run level, and then acts on these entries.

3.5.2 Changing Run Levels from Multiuser Mode

When the system is running at one of the two multiuser run levels, you can use the `init` command to change run levels. To use the command, log in as root and use the following syntax:

init [0|s|2|3|q]

The `init` command invokes the following run levels:

`0`	Specifies the halt state
`s`	Specifies the single-user run level
`2`	Specifies a multiuser run level with local processes and daemons
`3`	Specifies a multiuser run level with remote processes and daemons
`q`	Specifies that `init` should reexamine the `inittab` file

3.5.2.1 Changing to a Different Multiuser Run Level

To change from the current multiuser run level to a different multiuser run level, enter the `init` command with the argument that corresponds to the run level that you want to enter. For example, to change from run level 2 to run level 3, enter the following command:

```
# init 3
```

In response to your entry, `init` reads the `inittab` file and follows the instructions that correspond to the change in run level.

3.5.2.2 Changing to Single-User Mode

The `init` command provides a way to change from the current multiuser mode to single-user mode by using the `s` run level argument. For example, to change from the current run level to single-user mode, enter:

```
# init s
```

To change from a multiuser mode to single-user mode, giving users a 10-minute warning, enter:

```
# /usr/sbin/shutdown +10 Bringing system down to single-user for testing
```

To return to multiuser mode from single-user mode, use Ctrl/d or enter `exit` at the prompt. This causes the `init` command as process 1 to prompt you for the run level. In response to the prompt, enter `2` to return to multiuser mode without networking daemons activated, or enter `3` to return to multiuser mode with networking daemons activated.

Alternatively, you can reboot the system by using one of the following commands:

```
# /usr/sbin/shutdown -r now
```

```
# /sbin/reboot
```

3.5.2.3 Reexamining the inittab File

To reexamine the `inittab` file, enter the `init` command with the `q` argument, as follows:

```
# init q
```

In response, `init` reexamines the `inittab` file and starts new processes, if necessary. For example, if you recently added new terminal lines, `init` activates the `getty` process for these terminal lines in response to the `init q` command.

Refer to the `getty`(8) reference page for further information about the relationship between terminal lines and the `init` command.

3.6 Symmetric Multiprocessing

Symmetric MultiProcessing (SMP) consists of two or more processors that execute the same copy of the operating system, address common memory, and can execute instructions simultaneously. In a multiprocessor system, multiple threads can run concurrently through simultaneous execution on multiple processors.

If your system is a multiprocessor system and it is running Digital UNIX, it is running in an SMP environment. The objective of the operating system in an SMP environment is to take advantage of the incremental computes available to the system as additional processors are added. To do this, the operating system must allow multiple threads of execution to operate concurrently across the available processors.

From a system administrator's point of view, this additional computing power requires little to no additional system management work. All the administrator should see is additional available computes. It may be that additional I/O capabilities will be required to more efficiently utilize these extra computes.

3.6.1 Adding CPUs to an Existing System

At boot time, the system determines the number of CPUs available. Adding computing power to your multiprocessing systems is as simple as installing the processor board and rebooting the system. You do not have to reconfigure the kernel; you may have to modify any tuning that was done to limit the number of processors available, and you may need to install a Product Authorization Key (PAK). For more information on PAKs, see the *Software License Management* manual.

3.6.2 Unattended Reboots on Multiprocessor Systems

If a processor in a multiprocessor system fails, the operating system notes which processor failed, then automatically reboots the system. Although the operating system continues, you must manually restart the failed processor. For instructions, see the *Installation Guide*.

3.7 Setting and Resetting the System Clock

The system has an internal clock that you set when you install the system. The clock maintains the time and date whether the power is on or off. Nevertheless, there are occasions when you might need to reset the time or date. For example, with battery-powered clocks, you might need to reset the time as a result of battery failure; or you may need to synchronize system time with standard time.

To set the date and time, log in as root and use the following syntax with the `date` command:

date *[[cc]yy]mmddHHMM[.ss]*

cc	Designates the first two numbers of the year (century) as a 2-digit integer.
yy	Designates the year as a 2-digit integer
MM	Designates the month as a 2-digit integer
dd	Designates the day as a 2-digit integer
HH	Designates the hour as a 2-digit integer, using a 24-hour clock
mm	Designates the minutes as a 2-digit integer
.	Serves as a delimiter
ss	Designates the seconds as a 2-digit integer (this field is optional)

To set the date to 09:34:00 AM Jan 7, 2000 using the mmddHHMM[[cc]yy][.ss] Digital format:

```
# date  010709342000
# date  0107093400.00
# date  010709342000.00
```

Note

If you are changing the year, the system disk must be updated with the new year information. In single-user mode, enter the `mount –u /` command after you enter a date containing a new

year. This command writes the new year into the superblock on the system disk. Note also that the root file system will now be mounted read-write.

Refer to the `date`(1) reference page for more information.

3.8 Resolving Booting Problems

Should your system not boot, the following list suggests some areas for further investigation:

- Hardware failure

  Check the hardware manual accompanying your system for hardware test procedures. If a hardware problem exists, follow the instructions in the guide for resolving the problem.

- Software failure

  Software can fail for the following reasons:

  – An incorrect boot path was specified

    Refer to Section 3.3 or your system's hardware guide for instructions on specifying the correct boot path.

  – The kernel is corrupt

    If you suspect that the kernel may be corrupt, try booting the generic kernel, `/genvmunix`. This will provide you with a fully functional system and you can begin debugging procedures using the `kdbx` or `dbx` utilities to analyze crash dumps. Refer to the `kdbx`(8) or `dbx`(1) reference pages for more information. Refer to Section 3.3.1 for information on booting an alternate kernel.

  – A disk or file system is corrupt

    If a disk or file system is corrupt, run the `fsck` command on the file system. The `fsck` command checks and repairs UNIX File Systems (UFS). If `fsck` finds something wrong, it prompts you for an action to take. Use extreme care under these circumstances so that you do not inadvertently overwrite or remove any files. Refer to the `fsck`(8) reference page for more information. If you have an Advanced File System (AdvFS), disk corruption is very unlikely.

    AdvFS provides disk recovery during the mount procedure that corrects the disk structures. You do not need to run the `fsck` command or any other command. Consequently, recovery of AdvFS is very rapid.

3.9 Shutting Down the System

The following sections describe the shutdown procedures and the recovery strategies that you use in both controlled and unexpected shutdowns. The first part discusses procedures for handling controlled shutdowns. The second part discusses guidelines and recommendations for handling and recovering from unexpected shutdowns.

Note

You can also use the *SysMan* `dxshutdown` command for some of these tasks.

There are several good reasons to stop the system in a controlled shutdown. For example:

- You need to upgrade your software or add new hardware to your configuration. You shut down the system to set up the new additions, make the necessary adjustments to your configuration files, and build a new kernel.
- You have been monitoring the hardware error log and have noticed repeated warnings. You suspect that your hardware may soon fail so you shut down the system and examine the problem.
- You notice that system performance is degrading rapidly. You check the system statistics and conclude that some changes to the system would improve performance. You shut down and tune the system.
- You notice signs of possible file system corruption. You shut down the system and run the `fsck` program to fix problems or to confirm that none exist.

In each of these and similar situations a variety of options are available to you. Regardless of how you decide to resolve the situation, your first step is to initiate a controlled shutdown of the system. There are practical and reasonable ways to shut down your system from single-user mode or multiuser mode.

A system that has panicked or crashed presents you with a different set of circumstances than a system that has shut down in an orderly fashion. However, this chapter discusses orderly shutdowns only.

3.10 Stopping Systems While in Multiuser Mode

To shut down the system while running in multiuser mode, use the `shutdown` command. When you issue the `shutdown` command with the `-h` or `-r` flags, the program typically performs the following operations:

1. Runs the `wall` program to notify all users of the impending shutdown
2. Disables new logins
3. Stops all accounting and error-logging processes
4. Runs the `killall` program to stop all other processes
5. Runs the `sync` program to synchronize the disk(s)
6. Logs the shutdown in the log file
7. Unmounts file systems
8. Halts the system

The following sections describe typical shutdown operations and provide examples of what happens when you use the command flags. Refer to the `shutdown`(8) reference page for more information.

3.10.1 Shutting Down the System and Warning Other Users

To shut down the system from multiuser mode to single-user mode at a specific time and warn users of the shutdown:

1. Log in as root and change to the root directory:

   # **cd /**

2. Use the following syntax with the `shutdown` command:

   shutdown *time* [*warning-message*]

For example, to shut down the system in 10 minutes with a warning to users that the system is going down for routine maintenance tasks, enter:

shutdown +10 Maintenance shutdown

The system begins to notify users of the impending shutdown. Next, it disables logins, stops accounting and error logging, stops all remaining processes, logs the shutdown in the log file, and sends the `init` program a signal that causes the system to transition to single-user mode.

When the system's shutdown completion message appears, the shutdown is complete. You can access the system through the console to perform the desired administrative tasks. When you are finished, reboot the system.

3.10.2 Shutting Down and Halting the System

To shut down the system from multiuser mode, warn all users, and halt all systems:

1. Log in as root and change to the root directory:

   # **cd /**

2. Use the following syntax with the shutdown command:

   shutdown –**h** *time* [*warning-message*]

For example, to shut down and halt the system in 5 minutes with a warning to users that the system is going down for maintenance, enter:

```
# shutdown -h +5 Maintenance shutdown in five minutes
```

The shutdown program begins to notify users of the impending shutdown, disables logins, and proceeds with the standard shutdown activities. At the specified shutdown time, the systems are halted.

3.10.3 Shutting Down and Automatically Rebooting the System

To shut down the system from multiuser mode, warn all users, and automatically reboot the system to multiuser mode:

1. Log in as root and change to the root directory:

   # **cd /**

2. Use the following syntax with the shutdown command:

   shutdown -r *time* [*warning-message*]

For example, to shut down and automatically reboot the system in 15 minutes with a warning to users that the system is going down for a reboot, enter the following command:

```
# shutdown -r +15 Shutdown and reboot in 15 minutes
```

In this case, the system begins to notify users of the impending shutdown, disables logins, and proceeds with the standard shutdown activities. When it completes these activities, shutdown automatically starts the reboot operation, which involves running fsck for a consistency check on all mounted file systems. If problems are not encountered, the system reboots to multiuser mode.

Note

If fsck finds file system inconsistencies, it displays a warning message, recommending that you run fsck again from single-user mode before operating the system in multiuser mode.

3.10.4 Shutting Down and Halting Systems Immediately

To shut down and halt the system immediately:

1. Log in as root and change to the root directory. For example, enter the following command:

   # **cd /**

2. Enter the shutdown command as follows:

   # **shutdown -h now**

In response to this command, the system shuts down immediately and halts the system.

Note

Do not use this command when there are other users on the system. Users get no warning messages and their processes are immediately stopped.

3.11 Stopping Systems While in Single-User Mode

Although the shutdown command is your best choice for shutting down systems, you can also use the halt command. This command should be invoked only from single-user mode. If you are working in single-user mode, you can stop systems by entering the following commands:

```
# sync
# sync
# halt
```

In response to the halt command, the program logs the shutdown in the log file, kills all running processes, executes the sync system call and waits for all information to be written to disk, then halts the systems. Note that entering the sync command at least twice ensures that all data in memory is safely written to disk. Refer to the halt(8) reference page for a description of the command and its flags.

4

Customizing the System Environment

This chapter describes how you can customize your system environment in the following areas:

- System initialization files, which you use to initialize and control the system's run levels
- National language directories, which you use to supply support for language-specific and country-specific programs
- Internationalization features, which you tailor to support programmers and users developing and running programs for international audiences
- System time zone directories and envrionment variables, which you use to administer local and worldwide time zone information on your system
- System security tasks, which you employ to administer the security policy of your organization
- Performance monitors, which you set up and use to measure diverse aspects of system performance
- Power manager elements, which you set up and use to control power consumption in Energy Star-compliant peripherals and processors

4.1 Identifying and Modifying the System Initialization Files

To define and customize the system environment, you modify certain initialization files that specify and control processes and run levels. Digital UNIX provides you with default files that define the available run levels and the processes associated with each run level. You can easily change or customize the system environment by using these files as templates. In addition, if you support internationalization standards, you must be familiar with the structure and requirements of the corresponding files on your system.

This section describes the Digital UNIX software and provides instructions for identifying, using, and modifying the files that initialize and control the system environment. To understand and utilize available functionality, you should familiarize yourself with the `init` program and the specific files

and commands associated with the program. Refer to the `init`(8) reference page for a description of the program and its behavior.

Before you make any changes to the system initialization files, you should examine the default setup, evaluate the needs of your system, and make a copy of the entire set of default files. Taking precautions is wise when making changes to system files or to files that alter the working environment. If you discover that your modifications do not create the environment that you intended, you can reinstate the default files while you fix the problems in your customization.

The following system files and directories influence system startup and operation:

`/etc/inittab`
: One of the key initialization files whose entries define run levels and associated processes and administer terminals. Section 4.1.1 describes this file.

`/etc/securettys`
: A text file that marks whether a given tty line allows root logins. Section 4.1.1.7 describes this file.

`/sbin/bcheckrc`
: A system initialization run command script associated with checking and mounting file systems at startup time. Section 4.1.1.2 describes this file.

`/sbin/init.d`
: The initialization directory that contains executable files associated with system startup and the available run levels. Section 4.1.2.1 describes the directory structure and contents.

`/sbin/rc`*n* `.d`
: A set of individual directories that correspond to the various run levels. Each directory contains linked files that the system acts on when starting or changing a particular run level. There are three `/sbin/rc`*n* `.d` directories available: `/sbin/rc0.d`, `/sbin/rc2.d`, and `/sbin/rc3.d`. Section 4.1.2.2, Section 4.1.2.3, and Section 4.1.2.4 describe the `rc` directory structure and contents.

`/sbin/rc`*n*
: The run command script that corresponds to a particular run level. There are three `/sbin/rc`*n* scripts available: `/sbin/rc0`,

/sbin/rc2, and /sbin/rc3. Section 4.1.2.2, Section 4.1.2.3, and Section 4.1.2.4 describe the contents and use of these scripts.

/etc/rc.config
: A file that contains run-time configuration variables. Scripts in the /sbin/init.d directory use these variables to configure various subsystems (for example, NFS or NTP). You (or a program) can use the rcmgr command to define or access variables in the /etc/rc.config file. Refer to the rcmgr(8) reference page and the *Network Administration* manual for more information.

/etc/sysconfigtab
: The database file that contains information about the subsystems that can be dynamically configured. Chapter 5 describes this file.

/usr/sbin/getty
: The executable file that sets and manages terminal lines. Section 4.1.1.4 and Section 4.1.1.5 describe this program. Refer to the getty(8) reference page for more information.

/etc/gettydefs
: The file used by getty that contains entries to identify and define terminal line attributes. Refer to the gettydefs(4) reference page for more information.

/var/spool/cron/crontabs/*
: The files that contain entries to identify and define the regular or periodic activation of specific processes. Refer to Section 4.1.3 for more information about these files.

/var/spool/cron/atjobs/*
: The file that contains entries to identify and define the once-only activation of specific processes. See the at(1) reference page for more information.

The following files contain information on kernel configuration:

/usr/sys/conf/*NAME*
: The text file that defines the components that the system builds into your configuration. The *NAME* variable usually specifies the system name. Chapter 5 describes this file.

/usr/sys/conf/*NAME* .list
: The optional configuration file that stores information about the layered product subsystems and is used to automatically configure static subsystems. The *NAME* variable usually specifies the system name. Chapter 5 describes this file.

/usr/sys/conf/param.c
: The text file that contains default values for some tunable system parameters used in building the system's kernel. Chapter 5 describes this file.

4.1.1 Using the /etc/inittab File

One of the first actions taken by the `init` program is to read the `/etc/inittab` file. The `inittab` file supplies the `init` program with instructions for creating and running initialization processes. The `init` program reads the `inittab` file each time `init` is invoked. The file typically contains instructions for the default initialization, the creation and control of processes at each run level, and the `getty` line process that controls the activation of terminal lines.

The Digital UNIX software provides you with a basic `/etc/inittab` file that contains line entries for the most common and necessary initialization processes. For example, the `/etc/inittab` file available with the distribution software would look similar to the following:

```
is:3:initdefault:
ss:Ss:wait:/sbin/rc0 shutdown < /dev/console > /dev/console 2>&1
s0:0:wait:/sbin/rc0 off < /dev/console > /dev/console 2>&1
fs:23:wait:/sbin/bcheckrc < /dev/console > /dev/console 2>&1
# Dynamic loading not supported in this release.
#kls:23:bootwait:/sbin/kloadsrv < /dev/console > /dev/console 2>&1
#cfg:23:wait:/sbin/cfgmgr -l < /dev/console > /dev/console 2>&1
update:23:wait:/sbin/update > /dev/console 2>&1
it:23:wait:/sbin/it < /dev/console > /dev/console 2>&1
kmk:3:bootwait:/sbin/kmknod > /dev/console 2>&1
s2:23:wait:/sbin/rc2 < /dev/console > /dev/console 2>&1
s3:3:wait:/sbin/rc3 < /dev/console > /dev/console 2>&1
cons:1234:respawn:/usr/sbin/getty console console vt100
lat02:3:respawn:/usr/sbin/getty         /dev/tty02
lat03:3:respawn:/usr/sbin/getty         /dev/tty03
```

The `inittab` file is composed of an unlimited number of lines, each with four fields; each field is separated by a colon. The fields and syntax for entries in the `inittab` file are as follows:

Identifier: *Runlevel*: *Action*: *Command*

Identifier
: This 14-character field uniquely identifies an object entry.

Runlevel

This 20-character field defines the run levels in which the object entry is to be processed. The *Runlevel* variable corresponds to a configuration of processes in a system. Each process spawned by the `init` command is assigned one or more run levels in which it is allowed to exist. The run levels are as follows:

`0`	Specifies the halt state
`s` or `S`	Specifies single-user mode
`2`	Specifies multiuser mode without network services
`3`	Specifies multiuser mode with network services

The *Runlevel* field can define multiple run levels for a process by specifying more than one run level character in any combination.

Action

This 20-character field tells `init` how to treat the specified process. The most common actions that `init` recognizes are as follows:

`respawn`	If the process does not exist or dies, `init` starts it. If the process currently exists, `init` does nothing and continues scanning the `inittab` file.
`wait`	When `init` enters a run level that matches the run level of the entry, it starts the process and waits for its termination. As long as `init` continues in this run level, it does not act on subsequent reads of the entry in the `inittab` file.
`bootwait`	When `init` first executes and reads the `inittab` file, it processes this line entry. The `init` program starts the process, waits for its termination and, when it dies, does not restart the process.
`initdefault`	A line with this action is processed when `init` is first invoked. The `init` program uses this line to determine which run level to enter. To do this, it takes the highest run level specified in the run-level field and uses that as its initial state. If the run-level field is empty, this is interpreted as 0s23, so `init` enters run level 3. If `init` does not find an `initdefault` line in the `inittab` file, it requests an initial run level from the operator.

Other action keywords are available and recognized by the `init` program. See the `inittab`(4) reference page for more information.

Command
: This 1024-character field holds the sh command to be executed. The entry in the command field is prefixed with exec. Any legal sh syntax can appear in the command field.

You can insert comments in the inittab file by specifying a # (number sign) at the beginning of a line. You can also place a \ (line continuation character) at the end of a line.

If you intend to change or add entries to the /etc/inittab file, make certain that you are familiar with the function and contents of the associated files and run command scripts.

The following sections provide information that will help you to use the /etc/inittab file.

4.1.1.1 Specifying the Initialization Default Run Level

At boot time, the init program looks in the inittab file for the initdefault keyword to find the definition of the run level to enter. If there is no entry for initdefault, the system prompts you for a run level. In the previous inittab file example, the following line indicates that the run level for initdefault is set to 3, which is the multiuser with network services mode:

```
is:3:initdefault:
```

4.1.1.2 Specifying wait Run Levels

The init program looks in the inittab file for the wait entries. In the previous inittab file example, the following line contains a wait entry:

```
fs:23:wait:/sbin/bcheckrc < /dev/console > /dev/console 2>&1
```

In this case, the init program invokes the /sbin/bcheckrc script for the fs entry. Processes associated with this entry execute at run levels 2 and 3. Input comes from the system console (/dev/console). System and process error messages are sent to the console (> /dev/console 2>&1).

The bcheckrc run command script contains procedures associated with file system checking and mounting. See the /sbin/bcheckrc file for details.

4.1.1.3 Specifying bootwait Run Levels

The init program looks in the inittab file for the bootwait entry. In the previous inittab file example, the following line contains a bootwait entry:

```
kmk:3:bootwait:/sbin/kmknod > /dev/console 2>&1
```

In this case, the `init` program invokes the `/sbin/kmknod` script for the `kmk` entry.

4.1.1.4 Specifying Console Run Levels

Before you or anyone else can log in to your system, the `getty` program for nonworksystems and the `xdm` program for worksystems must set up the process that runs the login and shell programs for each terminal and workstation, respectively. Because a large portion of your initial work is done at the system console, the `/etc/inittab` file contains an entry for setting up a `getty` process for the console. The `xdm` process is started with a run-level script in the `/sbin/rc3.d` directory.

In the previous example of the `inittab` file, the following line contains the entry for the system console:

```
cons:1234:respawn:/usr/sbin/getty console console vt100
```

The `init` program is instructed to invoke the `getty` program, which sets the terminal line attributes for the system console (`/dev/console`). The run-level field specifies that the `getty` process executes at run levels 1, 2, 3, and 4. The `respawn` keyword tells `init` to re-create the `getty` process if the active process terminates. If the process is active, `init` does not respawn the process; if it terminates, the process is re-created.

Note

In general, you should not modify the system console entry in the `inittab` file unless you want to limit the system console's access to different run levels. By placing limitations on the range of run levels for this terminal line, you risk disabling the system console if the system enters a run level that prohibits execution of the console's `getty` process.

4.1.1.5 Specifying Terminals and Terminal Run Levels

To enable user logins at each terminal supported by your system, you must maintain support for the terminal types available at your site and define the run level and `getty` process for each supported terminal type. Use the following database and file:

- The `/usr/lib/terminfo` database (a symbolic link to `/usr/share/lib/terminfo`) defines the various terminal types.
- Entries in the `/etc/inittab` file define the run level and `getty` process for the supported terminal types.

The Digital UNIX system supports a wide variety of terminal types. The `terminfo` database contains entries that describe each terminal type and its capabilities. The database is created by the `tic` program, which compiles the source files into data files. The `terminfo` source files typically consist of at least one device description that conforms to a particular format. See the `terminfo`(4) reference page for specific details on creating and compiling source files.

The `/usr/lib/terminfo` directory contains the source files, each of which has a `.ti` suffix, for example *name*`.ti`. After you compile the source files with the `tic` command, it places the output in a directory subordinate to `/usr/lib/terminfo`.

Various commands and programs rely on the files in these directories. Set your TERMINFO environment variable to the `/usr/lib/terminfo` directory to instruct programs that rely on the database for information to look there for relevant terminal information.

See the `getty`(8), `gettydefs`(4), and `inittab`(4) reference pages for information about defining terminal lines and managing terminal access.

4.1.1.6 Specifying Process Run Levels

Specific entries in the `inittab` file define the run command scripts that are to be executed when the system enters or changes to a particular run level. For example, the following `inittab` file entries specify the action to be taken by the `init` program at each of the available run levels:

```
ss:Ss:wait:/sbin/rc0 shutdown < /dev/console > /dev/console 2>&1
s0:0:wait:/sbin/rc0 off < /dev/console > /dev/console 2>&1
s2:23:wait:/sbin/rc2 < /dev/console > /dev/console 2>&1
s3:3:wait:/sbin/rc3 < /dev/console > /dev/console 2>&1
```

These entries are associated with the `rc` directory structure and are discussed in detail in Section 4.1.2.

4.1.1.7 Securing a Terminal Line

The `/etc/securettys` file indicates to the system whether terminals or pseudoterminals can be used for root logins. To enable root logins on a terminal line, include the pathname in the `/etc/securettys` file. To enable root login on pseudoterminals, include the `ptys` keyword. You enable X displays for root login by including their display name, for example `:0`. By default, only the console and the X server line are set secure.

The following example of an `/etc/securettys` file shows root logins enabled on the console, on the X display, on two hard-wired or LAT lines, and on all pseudoterminals:

```
/dev/console
:0
/dev/tty00
/dev/tty01
ptys
```

4.1.2 Using the init and rc Directory Structure

The Digital UNIX system provides you with an initialization and run command directory structure. The structure has four main components: the `init.d`, `rc0.d`, `rc2.d`, and `rc3.d` directories. In addition, each of the `rc`*n*`.d` directories has a corresponding `rc`*n* run command script.

4.1.2.1 The init.d Directory

The `/sbin/init.d` directory contains the executable files associated with system initialization. For example, a listing of the directory contents would look similar to the following:

```
acct       inetd      motd       preserve    savecore   syslog
crashdc    kloadsrv   named      quota       sendmail   uucp
cron       kmod       nfs        recpasswd   settime    xdm
enlogin    lat        nfsmount   rmtmpfiles  sia        xntpd
gateway    loader     nis        route       snmpd
inet       lpd        paging     rwho        startlmf
```

4.1.2.2 The rc0.d Directory and rc0 Run Command Script

The `/sbin/rc0` script contains run commands that enable a smooth shutdown and bring the system to either a halt state or single-user mode. As described previously, the `inittab` file contains entries that the `init` program reads and acts on when the system is shutting down to single-user mode (level s) or halting (level 0). For example:

```
ss:Ss:wait:/sbin/rc0 shutdown < /dev/console > /dev/console 2>&1
s0:0:wait:/sbin/rc0 off < /dev/console > /dev/console 2>&1
```

Notice that in both cases, the `rc0` script is the specified command. In addition to commands listed in the script itself, `rc0` contains instructions to run commands found in the `/sbin/rc0.d` directory. These commands are linked to files in the `init.d` directory. The script defines the conditions under which the commands execute; some commands run if the system is being halted while others run if the system is being shut down and rebooted to single-user mode.

By convention, files in the `/sbin/rc0.d` directory begin with either the letter "K" or the letter "S" and are followed by a 2-digit number and a file

name. For example, a long listing of the rc0.d directory contents would look similar to the following:

```
lrwxr-xr-x 1 root staff 17 Jan 04 10:31 K00enlogin -> ../init.d/enlogin
lrwxr-xr-x 1 root staff 13 Jan 04 10:44 K05lpd -> ../init.d/lpd
lrwxr-xr-x 1 root staff 13 Jan 04 10:51 K07lat -> ../init.d/lat
lrwxr-xr-x 1 root staff 15 Jan 04 10:37 K10inetd -> ../init.d/inetd
lrwxr-xr-x 1 root staff 15 Jan 04 10:37 K15snmpd -> ../init.d/snmpd
lrwxr-xr-x 1 root staff 13 Jan 04 10:31 K19xdm -> ../init.d/xdm
lrwxr-xr-x 1 root staff 15 Jan 04 10:37 K20xntpd -> ../init.d/xntpd
lrwxr-xr-x 1 root staff 14 Jan 04 10:31 K22cron -> ../init.d/cron
lrwxr-xr-x 1 root staff 18 Jan 04 10:31 K25sendmail -> ../init.d/sendmail
lrwxr-xr-x 1 root staff 13 Jan 04 10:41 K30nfs -> ../init.d/nfs
lrwxr-xr-x 1 root staff 18 Jan 04 10:41 K35nfsmount -> ../init.d/nfsmount
lrwxr-xr-x 1 root staff 13 Jan 04 10:37 K38nis -> ../init.d/nis
lrwxr-xr-x 1 root staff 15 Jan 04 10:41 K40named -> ../init.d/named
lrwxr-xr-x 1 root staff 14 Jan 04 10:37 K42rwho -> ../init.d/rwho
lrwxr-xr-x 1 root staff 15 Jan 04 10:37 K43route -> ../init.d/route
lrwxr-xr-x 1 root staff 17 Jan 04 10:37 K44gateway -> ../init.d/gateway
lrwxr-xr-x 1 root staff 16 Jan 04 10:31 K45syslog -> ../init.d/syslog
lrwxr-xr-x 1 root staff 14 Jan 04 10:52 K46uucp -> ../init.d/uucp
lrwxr-xr-x 1 root staff 14 Jan 04 10:37 K50inet -> ../init.d/inet
lrwxr-xr-x 1 root staff 15 Jan 04 10:31 K52quota -> ../init.d/quota
```

In general, the system starts commands that begin with the letter "S" and stops commands that begin with the letter "K." The numbering of commands in the /sbin/rc0.d directory is important since the numbers are sorted and the commands are run in ascending order.

See the rc0(8) reference page for additional information.

4.1.2.3 The rc2.d Directory and rc2 Run Command Script

The /sbin/rc2 script contains run commands that enable initialization of the system to a nonnetworked multiuser state, run level 2. As described previously, the inittab file contains entries that the init program reads and acts on when the system is booting or changing its state to run level 2. For example:

```
s2:23:wait:/sbin/rc2 < /dev/console > /dev/console 2>&1
```

Notice that the rc2 script is the specified command. In addition to commands listed in the script itself, rc2 contains instructions to run commands found in the /sbin/rc2.d directory. These commands are linked to files in the init.d directory. The script defines the conditions under which the commands execute; some commands run if the system is booting, other commands run if the system is changing run levels.

By convention, files in the /sbin/rc2.d directory begin with either the letter "K" or the letter "S" and are followed by a 2-digit number and a file name. For example, a listing of the /sbin/rc2.d directory contents would look similar to the following:

```
lrwxr-xr-x 1 root staff 13 Jan 04 10:44 K00lpd -> ../init.d/lpd
lrwxr-xr-x 1 root staff 13 Jan 04 10:51 K03lat -> ../init.d/lat
```

```
lrwxr-xr-x 1 root staff 15 Jan 04 10:37 K05inetd -> ../init.d/inetd
lrwxr-xr-x 1 root staff 15 Jan 04 10:37 K10snmpd -> ../init.d/snmpd
lrwxr-xr-x 1 root staff 15 Jan 04 10:37 K15xntpd -> ../init.d/xntpd
lrwxr-xr-x 1 root staff 14 Jan 04 10:31 K20cron -> ../init.d/cron
lrwxr-xr-x 1 root staff 18 Jan 04 10:31 K30sendmail -> ../init.d/sendmail
lrwxr-xr-x 1 root staff 13 Jan 04 10:41 K35nfs -> ../init.d/nfs
lrwxr-xr-x 1 root staff 18 Jan 04 10:41 K40nfsmount -> ../init.d/nfsmount
lrwxr-xr-x 1 root staff 13 Jan 04 10:37 K43nis -> ../init.d/nis
lrwxr-xr-x 1 root staff 15 Jan 04 10:41 K45named -> ../init.d/named
lrwxr-xr-x 1 root staff 14 Jan 04 10:37 K47rwho -> ../init.d/rwho
lrwxr-xr-x 1 root staff 15 Jan 04 10:37 K48route -> ../init.d/route
lrwxr-xr-x 1 root staff 17 Jan 04 10:37 K49gateway -> ../init.d/gateway
lrwxr-xr-x 1 root staff 16 Jan 04 10:31 K50syslog -> ../init.d/syslog
lrwxr-xr-x 1 root staff 14 Jan 04 10:52 K51uucp -> ../init.d/uucp
lrwxr-xr-x 1 root staff 14 Jan 04 10:37 K55inet -> ../init.d/inet
lrwxr-xr-x 1 root staff 15 Jan 04 10:31 K57quota -> ../init.d/quota
lrwxr-xr-x 1 root staff 18 Jan 04 10:31 S00savecore -> ../init.d/savecore
lrwxr-xr-x 1 root staff 16 Jan 04 10:31 S05paging -> ../init.d/paging
lrwxr-xr-x 1 root staff 19 Jan 04 10:31 S10recpasswd -> ../init.d/recpasswd
lrwxr-xr-x 1 root staff 14 Jan 04 10:52 S15uucp -> ../init.d/uucp
lrwxr-xr-x 1 root staff 17 Jan 04 10:31 S25enlogin -> ../init.d/enlogin
```

In general, the system starts commands that begin with the letter "S" and stops commands that begin with the letter "K." Commands that begin with the letter "K" run only when the system is changing run levels from a higher to a lower level. Commands that begin with the letter "S" run in all cases. The numbering of commands in the `/sbin/rc2.d` directory is important since the numbers are sorted and the commands are run in ascending order.

Refer to the `rc2`(8) reference page for more information.

4.1.2.4 The rc3.d Directory and rc3 Run Command Script

The `/sbin/rc3` script contains run commands that enable initialization of the system to a networked multiuser state, run level 3. As described previously, the `inittab` file contains entries that the `init` program reads and acts on when the system is booting or changing its state to run level 3. For example:

```
s3:3:wait:/sbin/rc3 < /dev/console > /dev/console 2>&1
```

Notice that the `rc3` script is the specified command. In addition to commands listed in the script itself, `rc3` contains instructions to run commands found in the `/sbin/rc3.d` directory. These commands are linked to files in the `init.d` directory. The script defines the conditions under which the commands execute; some commands run if the system is booting, other commands run if the system is changing run levels.

By convention, files in the `/sbin/rc3.d` directory begin with the letter "S" and are followed by a 2-digit number and a file name. For example, a long listing of the `rc3.d` directory contents would look similar to the following:

```
lrwxr-xr-x 1 root staff 14 Jan 04 10:37 S00inet -> ../init.d/inet
lrwxr-xr-x 1 root staff 15 Jan 04 10:31 S01quota -> ../init.d/quota
```

```
lrwxr-xr-x 1 root staff 14 Jan 04 10:52 S04uucp -> ../init.d/uucp
lrwxr-xr-x 1 root staff 17 Jan 04 10:31 S05settime -> ../init.d/settime
lrwxr-xr-x 1 root staff 18 Jan 04 10:31 S08startlmf -> ../init.d/startlmf
lrwxr-xr-x 1 root staff 16 Jan 04 10:31 S10syslog -> ../init.d/syslog
lrwxr-xr-x 1 root staff 17 Jan 04 10:37 S11gateway -> ../init.d/gateway
lrwxr-xr-x 1 root staff 15 Jan 04 10:37 S12route -> ../init.d/route
lrwxr-xr-x 1 root staff 14 Jan 04 10:37 S13rwho -> ../init.d/rwho
lrwxr-xr-x 1 root staff 15 Jan 04 10:41 S15named -> ../init.d/named
lrwxr-xr-x 1 root staff 13 Jan 04 10:37 S18nis -> ../init.d/nis
lrwxr-xr-x 1 root staff 18 Jan 04 10:41 S20nfsmount -> ../init.d/nfsmount
lrwxr-xr-x 1 root staff 16 Jan 04 10:31 S22loader -> ../init.d/loader
lrwxr-xr-x 1 root staff 18 Jan 04 10:31 S23kloadsrv -> ../init.d/kloadsrv
lrwxr-xr-x 1 root staff 14 Jan 04 10:31 S24kmod -> ../init.d/kmod
lrwxr-xr-x 1 root staff 18 Jan 04 10:31 S25preserve -> ../init.d/preserve
lrwxr-xr-x 1 root staff 13 Jan 04 10:31 S26sia -> ../init.d/sia
lrwxr-xr-x 1 root staff 20 Jan 04 10:31 S30rmtmpfiles -> ../init.d/rmtmpfiles
lrwxr-xr-x 1 root staff 13 Jan 04 10:41 S35nfs -> ../init.d/nfs
lrwxr-xr-x 1 root staff 18 Jan 04 10:31 S40sendmail -> ../init.d/sendmail
lrwxr-xr-x 1 root staff 15 Jan 04 10:37 S45xntpd -> ../init.d/xntpd
lrwxr-xr-x 1 root staff 15 Jan 04 10:37 S50snmpd -> ../init.d/snmpd
lrwxr-xr-x 1 root staff 15 Jan 04 10:37 S55inetd -> ../init.d/inetd
lrwxr-xr-x 1 root staff 14 Jan 04 10:31 S57cron -> ../init.d/cron
lrwxr-xr-x 1 root staff 13 Jan 04 10:30 S58lat -> ../init.d/lat
lrwxr-xr-x 1 root staff 14 Jan 04 10:30 S60motd -> ../init.d/motd
lrwxr-xr-x 1 root staff 13 Jan 04 10:44 S65lpd -> ../init.d/lpd
lrwxr-xr-x 1 root staff 14 Jan 04 10:42 S75acct -> ../init.d/acct
lrwxr-xr-x 1 root staff 17 Jan 04 10:30 S80crashdc -> ../init.d/crashdc
lrwxr-xr-x 1 root staff 13 Jan 04 10:30 S95xdm -> ../init.d/xdm
```

In general, the system starts commands that begin with the letter "S" and stops commands that begin with the letter "K." Commands that begin with the letter "K" run only when the system is changing run levels from a higher to a lower level. Commands that begin with the letter "S" run in all cases.

Usually, only commands that begin with the letter "S" are placed in the `rc3.d` directory. By default, run level 3 is the highest run level. The numbering of commands in the `/sbin/rc3.d` directory is important since the numbers are sorted and the commands are run in ascending order.

Refer to the `rc3`(8) reference page for more information.

4.1.3 Using the crontabs Directory

The `crontab` command submits a schedule of commands to the `cron` system clock daemon. The `cron` daemon runs shell commands according to the dates and times specified in the files in the `/var/spool/cron/crontabs` directory. Commands that you want to run on a regular schedule are in these files. Commands that you want to run only once are in the `/var/spool/cron/atjobs/*` files and are submitted with the `at` command.

The following example of an entry from a file in the `/var/spool/cron/crontabs` directory specifies that the `runacct`

command runs at 2:00 a.m., Monday through Saturday, and output is sent to the /var/adm/acct/nite/fd2log file:

```
  1                  2                        3
0 2 * * 1-6 /usr/sbin/acct/runacct > /var/adm/acct/nite/fd2log&
```

Each entry has the following syntax:

1. Specifies the minutes past the hour, the hour, day of month, month, and day of week. Note that for the day of week, the value 0 (zero) indicates Sunday, the value 1 indicates Monday, and so on. You can specify a single value, more than one value separated by commas, or two values separated by a dash (–) to indicate a range of values. You can also specify an asterisk (*) to indicate no specific value. For example, if an asterisk (*) is specified for the hour, the command is run every hour.
2. Specifies the command to be executed at the specified time.
3. Specifies, optionally, arguments to the command.

To add a comment to a file, specify a # (number sign) at the beginning of the line.

The files in the /var/spool/cron/crontabs directory are named for system users, and the commands in the files are run under the authority of the user. For example, the commands in the adm file are run under adm authority.

To use the crontab command, you must be the user that matches the file name you want to act upon. For example, if you are user adm and you run the crontab command, the action is performed on the /var/spool/cron/crontabs/adm file.

To submit commands to the cron daemon to be run under adm authority:

1. Become user adm.
2. Enter the crontab command with the –l option to copy the /usr/spool/cron/crontabs/adm file to a temporary file in your home directory.

   ```
   % crontab –l > temp_adm
   ```

3. Edit the temporary file and add the commands you want to run at a specified time.
4. Enter the crontab command and specify the temporary file to submit the commands to the cron daemon.

   ```
   % crontab temp_adm
   ```

The /var/adm/cron/log file contains a history of the commands executed by the cron daemon. This file should be monitored to prevent it from becoming excessively large.

Refer to the crontab(1) reference page for more information.

4.2 Identifying and Managing National Language Support Directories and Files

Digital UNIX provides language-specific and country-specific information or support for programs.

The support components that concern you most directly as system administrator are the directories and files that reside at /usr/lib/nls.

An internationalized system presents information in a variety of ways. The word "locale" refers to the language, territory, and code set requirements that correspond to a particular part of the world. The system stores locale-specific data in two kinds of files:

- Locale files, which contain month and day names, date formats, monetary and numeric formats, valid yes/no strings, character classification data, and collation sequences. These files reside in the /usr/lib/nls/loc directory.
- Message catalogs, which contain translations of messages used by programs. These files reside in the /usr/lib/nls/msg/*locale-name* directory.

Table 4–1 lists the locales moved to the /usr/lib/nls/loc directory when you install the optional Single-Byte European Locales subset. Additional locales are installed by language variant subsets with special licensing requirements.

Table 4–1: Locale Support Files

Language/Territory	Locale Filename
Danish-Denmark	da_DK.ISO8859-1
Dutch-Netherlands	nl_NL.ISO8859-1
Dutch_Belgium	nl_BE.ISO8859-1
English_U.K	en_GB.ISO8859-1
English_U.S.A.	en_US.ISO8859-1
Finnish-Finland	fi_FI.ISO8859-1

Table 4–1: Locale Support Files (cont.)

Language/Territory	Locale Filename
French_Belgium	fr_BE.ISO8859-1
French_Canada	fr_CA.ISO8859-1
French_France	fr_FR.ISO8859-1
French_Switzerland	fr_CH.ISO8859-1
German_Belgium	de_BE.ISO8859-1
German_Germany	de_DE.ISO8859-1
German_Switzerland	de_CH.ISO8859-1
Greek-Greece	el_GR.ISO8859-7
Italian-Italy	it_IT.ISO8859-1
Norwegian-Norway	no_NO.ISO8859-1
Portuguese-Portugal	pt_PT.ISO8859-1
Spanish-Spain	es_ES.ISO8859-1
Swedish-Sweden	sv_SV.ISO8859-1
Turkish-Turkey	tr_TR.ISO8859-1

Note

The `/usr/lib/nls/loc` directory also contains environment tables (.en files), character tables (.8859* files), and DEC variants (@DEC files) that correspond to some of the files listed in Table 4–1. These tables and variants are provided only to ensure system compatibility for old programs and should not be used by new applications.

4.2.1 Setting Locale

The default system-wide locale for internationalization is the C locale. The default system-wide locale is the one that the `setlocale` function uses when a user does not set the internationalization environment variables, such as LANG, LC_COLLATE, and so on.

To change the system-wide default locale for Bourne and Korn shell users, edit the `/etc/profile` file and include the name of the locale you want to be the system-wide default. The `setlocale` function will then use the

locale specified in this file. Those using the C shell can set a system-wide locale by editing the /etc/csh.login file and including the name of the locale you want to be the default system-wide locale.

You can set the native locale to any of the locales in the /usr/lib/nls/loc directory.

To set a locale, assign a locale name to one or more environment variables in the appropriate shell startup file. The simplest way is to assign a value to the LANG environment variable because it covers all components of a locale.

Note

The C locale mentioned in Table 4–1 is the system default. The C locale specifies U.S. English and uses the 7-bit ASCII codeset. The main difference between the C locale and the U.S. English locale (en_US.ISO8859-1) is that the latter has enhanced error messages.

The following example sets the locale to French for the C shell in which it is invoked and for all child processes of that shell:

```
% setenv LANG fr_FR.ISO8859-1
```

If you want another shell to have a different locale, you can reset the LANG environment variable in that particular shell. The following example sets the locale to French for the Korn and Bourne shells:

```
$ LANG=fr_FR.ISO8859-1
$ export LANG
```

Note that setting the LANG environment variable on the command line sets the locale for the current process only.

In most cases, assigning a value to the LANG environment variable is the only thing you need to do to set the locale. This is because when you set the locale with the LANG environment variable, the appropriate defaults are automatically set for the following functions:

- Collation
- Character classification
- Date and time conventions
- Numeric and monetary formats
- Program messages
- Yes/no prompts

In the unlikely event that you need to change the default behavior of any of the previous categories within a locale, you can set the variable that is associated with that category. See the following section for more information.

4.2.2 Modifying Locale Categories

When you set the locale with the LANG environment variable, defaults are automatically set for the collation sequence, character classification functions, date and time conventions, numeric and monetary formats, program messages, and the yes/no prompts appropriate for that locale. However, should you need to change any of the default categories, you can set the environment variables that are associated with one or more categories.

Table 4–2 describes the environment variables that influence locale categories.

Table 4–2: Locale Environment Variables

Environment Variable	Description
LC_ALL	Overrides the setting of all other internationalization environment variables, including LANG.
LC_COLLATE	Specifies the collating sequence to use when sorting names and when character ranges occur in patterns.
LC_CTYPE	Specifies the character classification information to use.
LC_NUMERIC	Specifies the numeric format.
LC_MONETARY	Specifies the monetary format.
LC_TIME	Specifies the date and time format.
LC_MESSAGES	Specifies the language in which system messages will appear. In addition, specifies the strings that indicate "yes" and "no" in yes/no prompts.

As with the LANG environment variable, you can assign all of the category variables locale names. For example, suppose that your company's main language is Spanish. You can set the locale with the LANG environment variable for Spanish, but set the numeric and monetary format for U.S. English. To do this for the C shell, you would make the following variable assignments:

```
% setenv LANG es_ES.ISO8859-1
% setenv LC_NUMERIC en_US.ISO8859-1
% setenv LC_MONETARY en_US.ISO8859-1
```

Locale names may include *@modifiers* to indicate versions of the locales that meet special requirements for different categories.

For example, a locale might exist in two versions to sort data two ways: in dictionary order and in telephone-book order. Suppose your site is in France, uses the default French locale, and the standard setup for this locale uses dictionary order. However, your site also needs to use a site-defined locale that collates data in telephone-book order. You might set your environment variables for the C shell as follows:

```
% setenv LANG fr_FR.ISO8859-1
% setenv LC_COLLATE fr_FR.ISO8859-1@phone
```

The explicit setting of LC_COLLATE overrides LANG's implicit setting of that portion of the locale.

4.2.3 Limitations of Locale Variables

The LANG and LC_* environment variables allow you to set the locale the way you want it, but they do not protect you from mistakes. There is nothing to protect you from setting LANG to a Swedish locale and LC_CTYPE to a Portuguese locale.

Also, there is no way to tie locale information to data. This means that the system has no way of knowing what locale you set when you created a file, and it does not prevent you from processing that data in inappropriate ways later. For example, suppose LANG was set to a German locale when you created file `foo`. Now suppose you reset LANG to a Spanish locale and then use the `grep` command for something in `foo`. The `grep` command will use Spanish rules on the German data in the file.

4.2.4 Setting Environment Variables for Message Catalogs and Locales

To define the location of message catalogs, set the NLSPATH environment variable. The default path is as follows:

```
NLSPATH=/usr/lib/nls/msg/%L/%N:
```

In this example, `%L` specifies the current locale name, and `%N` specifies the value of name of the message catalog.

There is also a LOCPATH environment variable that defines the search path for locales. The default path is as follows:

```
LOCPATH=/usr/lib/nls/loc:
```

4.3 Customizing Internationalization Features

Digital UNIX is an internationalized operating system. Your site's planners determine which elements of the operating system's internationalization features, commonly called worldwide support features, are required. The worldwide support features are optional subsets that you can select during installation. Your job as an administrator is to set up and maintain these features for:

- Software developers who produce internationalized applications
- Users who run internationalized applications on your system

The Digital UNIX product provides three sources of information about worldwide support:

- For a list of optional software subsets that support internationalization, see the Digital UNIX *Installation Guide*.
- For information about setting up and maintaining an operating system environment for programmers who write internationalized software, see the Digital UNIX manual *Writing Software for the International Market*.
- To set up and maintain your system for users of internationalized applications, see the *SysMan* graphical interface and click on the Configuration icon and then the internationalization icon. From the internationalization window, you can select tasks to configure or modify several of the worldwide support capabilities on your system.

4.4 Customizing Your Time Zone

Information about configuring your system's time zone is in Chapter 5. This section describes how to administer local and worldwide time zone information on your system.

Time zone information is stored in files in the `/etc/zoneinfo` directory. The `/etc/zoneinfo/localtime` file is linked to a file in the `/etc/zoneinfo` directory and specifies the local time zone. These files are linked during system installation, but, as superuser, you can change your local time zone by relinking the `/etc/zoneinfo/localtime` file. For example, the following command changes the local time zone to Canada's Atlantic time zone:

```
# ln -sf /etc/zoneinfo/Canada/Atlantic /etc/zoneinfo/localtime
```

The `/etc/zoneinfo/sources` directory contains source files that specify the worldwide time zone and daylight savings time information that is used to generate the files in the `/etc/zoneinfo` directory. You can change the information in the source files and then use the `zic` command to

generate a new file in the /etc/zoneinfo directory. Refer to the zic(8) reference page for more information.

You can also change the default time zone information by setting the TZ environment variable in your .login file or shell environment file. If you define the TZ environment variable, its value overrides the default time zone information specified by /etc/zoneinfo/localtime. By default, the TZ variable is not defined.

The TZ environment variable has the following syntax:

stdoffset [*dst*[offset] [,*start*[/*time*], *end*[/*time*]]]

You can also specify the following syntax:

stdoffset [*dst*[*offset*]]

The TZ environment variable syntaxes have the following parameters:

std and *dst*

Specifies the three or more characters that designate the standard (*std*) or daylight savings time (*dst*) zone.

Note

Daylight savings time is called daylight summer time in some locales.

The *dst* variable is not specified, daylight savings time time does not apply. You can specify any uppercase and lowercase letters. A leading colon (:), comma (,), hyphen (-), plus sign(+), and ASCII NUL are not allowed.

offset

Specifies the value to be added to the local time to arrive at GMT. The *offset* variable uses 24-hour time and has the following syntax:

hh [:*mm* [:*ss*]]

If you do not specify the *offset* variable after the *dst* variable, daylight savings time is assumed to be 1 hour ahead of standard time. You can specify a minus sign (–) before the *offset* variable to indicate that the time zone is east of the prime meridian; west is the default, which you can specify with a plus sign (+).

start and *end*

Specifies when daylight savings time starts and ends. The *start* and *end* variable has the following syntaxes:

```
Jj
n

Mm.w.d
```

In the first syntax, the *j* variable specifies the Julian day, which is between 1 and 365. The extra day in a leap year (February 29) is not counted.

In the second syntax, the *n* variable specifies the zero-based Julian day, which is between zero (0) and 365. The extra day in a leap year is counted.

In the third syntax, the *m* variable specifies the month number (from 1 to 12), the *w* variable specifies the week number (from 1 to 5), and the *d* variable specifies the day of the week (from 0 to 6), where zero (0) specifies Sunday and six (6) specifies Saturday.

time

Specifies the time, in local time, when the change occurs to or from daylight savings time. The *time* variable uses 24-hour time and has the following syntax:
hh [:*mm* [:*ss*]]

The default is 02:00:00.

The following example of the TZ environment variable specification specifies:

- EST (eastern standard time) specifies the standard time, which is 5 hours behind GMT.
- EDT (eastern daylight time) specifies the daylight savings time, which is 4 hours behind GMT.
- EDT starts on the first Sunday in April and ends on the last Sunday in October; the change to and from daylight savings time occurs at 2:00, which is the default time.

```
EST5EDT4,M4.1.0,M10.5.0
```

You can also specify the following syntax:

:*pathname*

The *pathname* variable specifies the pathname of a file that is in the `tzfile` file format and that contains the time conversion information. For example:

```
:US/Eastern
```

Refer to the `tzfile`(4) reference page for more information on the file format.

If the pathname begins with a slash (/), it specifies an absolute pathname; otherwise, the pathname is relative to the `/etc/zoneinfo` directory. If the specified file is unavailable or corrupted, the system defaults to the offset stored in the kernel `tz` structure.

The time zone formats differ for SVID 2 and SVID 3. For SVID 2, `/usr/sbin/timezone` creates the `/etc/svid2_tz` file. The contents of the TZ and TZC variables are based on the information you supply when you run `/usr/sbin/timezone`.

For SVID 3, the `/etc/svid3_tz` file is created during the installation process. The contents of the TZ variable is based upon answers you supply to time zone-related questions at installation time.

Refer to the `timezone`(3) reference page for more information.

Refer to Chapter 5 for information about configuring a time zone for your system.

4.5 Customizing System Security

The system security tasks of the administrator range from the protection of physical components of the system and its environment to the implementation of an organization's security policies.

Two manuals in the Digital UNIX documentation set describe security-related tasks. Refer to the following documents for information about administering local system security:

- The *Technical Overview* briefly describes the security components of the Digital UNIX operating system.
- The *Security* manual is the principal source of security-related information for Digital UNIX users, administrators, and programmers dealing with the security components. Use this manual to administer security on an Digital UNIX operating system.

4.6 Customizing Performance Monitors

This section discusses how to set up and use some of the performance monitoring components of the Digital UNIX operating system:

- Monitoring Performance History
- Performance Monitor
- Digital UNIX performance monitoring commands and scripts

4.6.1 Monitoring Performance History Utility

The Monitoring Performance History (MPH) utility gathers timely and accurate information on the reliability and availability of the Digital UNIX operating system and its hardware environment.

MPH is a suite of shell scripts that copy error log and crash dump information twice per week. The information is automatically copied to Digital for analysis via Internet Mail. After analysis, reports are generated and distributed to the users of this information, namely Software and Hardware Engineering, Manufacturing, and Digital Services. This data is internally secure to Digital and will be used exclusively for monitoring purposes.

The MPH process is automatic, requiring no human intervention and no training. The installation time is approximately 10 minutes.

This software will not impact or degrade your system's performance. MPH runs as a background task, using very negligible CPU resources and is invisible to the user. The disk space required for the collected data and the application is approximately 300 blocks per system. This could be slightly higher in the case of a high number of errors.

Before running MPH, review the following information:

- The Software Development Environment subset (OSFPGMR) must be installed.
- The MPH software kit is contained in the mandatory base software subset OSFHWBASE400. This subset is installed automatically during the operating system installation.
- Disk space requirements for MPH is approximately 100 blocks.
- If the operating system needs to be shut down for any reason, an orderly shutdown process must be followed. Otherwise, you will have to restart the MPH script.

To run MPH on your system, complete the following steps:

1. Enter the following command to run the MPH script:

   # **MPH_OSF_018.CSH**

2. Enter the information requested by the script.

Running the MPH_OSF_018.CSH script does the following:

- Creates an MPH directory. The default directory location is /var/mph.
- Updates the system crontab files to execute the MPH files at the appropriate times. The binary error log extractor runs daily at 2:00 a.m. The data is mailed to Digital at 3:00 a.m. every Wednesday and Sunday.

4.6.2 Performance Monitor

The Performance Monitor is a real-time performance monitor that allows you to detect and correct performance problems. You can display graphs and counters to monitor dozens of different system values, including CPU performance, memory usage, disk transfers, file-system capacity, network efficiency, and buffer cache hit rates. In addition, thresholds can be set to alert you to, or correct, a problem when it occurs, and archives of data can be kept for high-speed playback or compression into charts, showing resource usage trends.

The Performance Monitor is an optional subset in the Digital UNIX software kit. For information about establishing and using the Performance Monitor, see the *Performance Monitor User's Guide*.

4.6.3 Performance Manager

Performance Manager is a real-time performance monitor that allows users to detect and correct performance problems. Graphs and charts can show hundreds of different system values, including CPU performance, memory usage, disk transfers, file- system capacity, and network efficiency. Thresholds can be set to alert you to correct a problem when it occurs, commands can be run on multiple nodes from the graphical user interface, and archives of data can be kept for high-speed playback or long-term trend analysis. See the *Installation Guide* for information about this product.

4.6.4 UNIX Commands and Scripts

Many Digital UNIX commands and scripts can be used to establish and use good system monitoring practices. For information about these commands

and scripts, see the Digital UNIX *System Tuning and Performance Management* manual.

4.7 Customizing Power Management

Use the `dxpower` utility, the `sysconfig` command, and `sysconfigdb` database to manage power-saving features on hardware subsystems, such as processors and peripherals, that employ power managment capabilities. With these utilities, you enable power management modes and specify the amount of time to wait before shutting off each component in order to save power.

4.7.1 Using the dxpower Utility's Graphical User Interface

If you have CDE installed on your system, you can open the `dxpower` power management utility by performing the following steps:

1. Click on the Application Manager icon.
2. Double click on the System_Admin application group icon.
3. Double click on the DailyAdmin application group icon.
4. Double click on the Power Management icon.

If you are not using CDE, you can start the `dxpower` utility from the command line as follows:

```
# /usr/bin/X11/dxpower
```

When the `dxpower` utility runs, a power management window is displayed on your screen. The window provides check boxes that you use to select modes of operation, and scales you use to specify dwell times.

For more information about how to use the `dxpower` utility, start the application and then click on the Help button in the lower right-hand corner of the window.

4.7.2 Implementing Power Management from the Command Line

You can control power management attributes from the command line by using `sysconfig` commands to manage the `sysconfigdb` database. For example, you will need to use these commands if you are activating power management for a system from a remote terminal or from a local console terminal.

If you activate the power management tools from a console terminal where CDE is not running, only the `graphics_powerdown` and

graphics_off_dwell attributes apply. Changing the graphics_standby_dwell and graphics_suspend_dwell attribute values has no effect. See Section 4.7.2.1 for descriptions of these attributes.

Caution

Do not attempt to use the sysconfig commands and dxpower simultaneously. If you do, you could encounter unpredictable behavior.

4.7.2.1 Changing the Power Management Values

To change the power management values that take effect every time you restart the kernel, you create a file in stanza file format. See stanza(4) for more information. The stanza-formatted file can contain the following power management attributes:

- default_pwrmgr_state

  The global power management state. Specify 1 to enable or 0 to disable this attribute.

- cpu_slowdown

  The current state of CPU slowdown. Specify 1 to enable or 0 to disable this attribute.

- disk_dwell_time

  The default dwell time, in minutes, for registered disks.

- disk_spindown

  The current state of disk spindown. Specify 1 to enable or 0 tor disable this attribute.

- graphics_powerdown

  The current state of graphics powerdown. Specify 1 to enable or 0 to disable this attribute.

- graphics_standby_dwell

  The default dwell time, in minutes, for standby Display power management Signaling (DPMS) mode. Specify a value of 0 to disable this attribute.

- graphics_suspend_dwell

  The default dwell time, in minutes, for suspend DPMS mode. Specify 0 to disable this attribute or specify a value greater than or equal to the value for graphics_standby_dwell.

- graphics_off_dwell

  The default dwell time, in minutes, for off DPMS mode. Specify 0 to disable this attribute or specify a value greater than or equal to the values for graphics_standby_dwell and graphics_suspend_dwell.

For example, you can create a stanza file called power_mgr.stanza that defines the following values for the attributes:

```
pwrmgr:
  default_pwrmgr_state=1
  cpu_slowdown=1
  disk_dwell_time=20
  disk_spindown=1
  graphics_powerdown=1
  graphics_standby_dwell=5
  graphics_suspend_dwell=10
  graphics_off_dwell=15
```

For the disk_dwell_time, graphics_standby_dwell, graphics_suspend_dwell, and graphics_off_dwell attributes, the specified values indicate the number of minutes to wait before powering down the idle hardware. In this case, the power management subsystem waits 20 minutes before disk spindown, and 5, 10, and 15 minutes before DPMS standby, suspend, and off modes, respectively. The remaining attributes, have a value of 1, which indicates that the function is enabled.

After you create and save the stanza file, enter the following command to update the /etc/sysconfigtab database:

```
# sysconfigdb -a -f power_mgr.stanza pwrmgr
```

See the sysconfigdb(8) reference page for more information.

4.7.2.2 Changing a Running Kernel or X Server

To change the values of attributes in the running kernel, use the sysconfig –r command. For example:

```
# sysconfig -r pwrmgr cpu_slowdown=0
```

You can change more than one attribute at a time, as shown in the following example:

```
# sysconfig -r pwrmgr graphics_powerdown=1 graphics_standby_dwell=10
```

See the sysconfig(8) reference page for more information.

See the dpms switches in the Xdec(1X) and xset(1X) reference pages for information about changing Display power management Signalling modes and values in the X Server.

5

Configuring the Kernel

The Digital UNIX kernel is a memory-resident executable image that handles all the system services —– hardware interrupts, memory management, interprocess communication, process scheduling —– and makes all other work on the operating system possible. In addition to the code that supports these core services, the kernel contains a number of subsystems.

A subsystem is a kernel module that extends the kernel beyond the core kernel services. File systems, network protocol families, and physical and psuedodevice drivers are all examples of supported subsystems. Some subsystems are required in the kernel, while others are optional. You configure your kernel by adding and removing these optional subsystems.

You also configure your kernel by tuning certain values stored in it. For example, the kernel contains values that can be adjusted to help make disk access faster. Modifying those values to optimize disk access can improve your system's performance.

The system provides two methods of configuring your kernel: the dynamic method and the static method. Dynamic system configuration entails using commands to configure the kernel. Static system configuration entails modifying system files and rebuilding the kernel. Modifying system files and rebuilding the kernel can be a difficult process, so use dynamic kernel configuration whenever possible.

This chapter helps you understand kernel configuration by explaining the following:

- How the kernel is configured at installation time
- How to determine whether you need to configure your kernel and which configuration method to use, static or dynamic
- How to configure your system dynamically, using commands
- How to configure your system statically, by editing system files and rebuilding the kernel

5.1 System Configuration at Installation Time

When you install Digital UNIX, the installation program initially copies a kernel image to the root partition of your system disk. This kernel image, known as the generic kernel, supports all processors and hardware options that are available for use with the current version of the operating system. In this way, the installation program ensures that you can boot your system regardless of its configuration.

Towards the end of the installation, after all the subsets you selected have been written to disk and verified, the installation program calls the `/usr/sbin/doconfig` program. As part of its processing, the `/usr/sbin/doconfig` program calls another program, known as the `sizer` program. The `sizer` program determines what hardware and software options are installed on your system and builds a target configuration file specific to your system. (The configuration file is the system file that controls what hardware and software support is linked into the kernel.) The `/usr/sbin/doconfig` program then builds your target kernel from this target configuration file.

Unlike the generic kernel copied to the system at installation time, the target kernel is tailored to your system. Only the hardware and software options available on your system are compiled into the target kernel. As a result, the target kernel is much smaller than the generic kernel.

When the installation is complete, the target kernel resides either in the root partition of your system disk or in memory, depending upon how your system was built. (See Section 5.4 for information about the different ways in which your kernel can be built.) If the appropriate console boot variables are set, your system always boots the target kernel automatically. (For information about setting console boot variables, see Chapter 3.)

5.2 Deciding When and How to Reconfigure Your Kernel

After your target kernel is built and started by the installation procedure, you can use it without modifications, unless one of the following occurs:

- You decide to add new subsystems to the kernel, for example by installing a new device.
- You decide to remove subsystems from the kernel, for example by removing a device.
- The performance of your system is poor, so you decide to tune values in the kernel. These values are called subsystem attributes if they are dynamically configurable or system parameters if they are statically configurable.

You must reconfigure your kernel, either dynamically or statically, when one of these situations occurs. The method you use to reconfigure your kernel depends upon the support provided by the subsystem or subsystem attributes.

Some kernel subsystems, such as the `decnet` subsystem, are dynamically loadable, meaning that you can add the subsystem to or remove the subsystem from the kernel without rebuilding the kernel. Often, subsystems that are dynamically loadable also allow you to dynamically configure the value of their attributes. Therefore, you can tune the performance of these subsystems without rebuilding the kernel. The following subsystems provided by Digital are dynamically loadable and allow dynamic configuration of attributes:

- `decnet` (DECnet network software)
- `lat` (Local Area Terminal)

If you decide to add or remove these subsystems from the kernel or configure the value of their attributes, use the procedures described in Section 5.3.

Some subsystems, such as required subsystems, are not dynamically loadable. However, these subsystems might allow you to dynamically configure the value of attributes. If so, you can configure the value of these subsystem attributes without rebuilding the kernel.

Digital UNIX offers two methods of dynamically configuring attributes:

- You can configure the value of attributes in the currently running kernel using the `sysconfig -r` command. Only a few kernel subsystems support this run-time configuration.
- You can configure the value of attributes in the dynamic subsystem database, `/etc/sysconfigtab`. When you want to begin running a kernel that contains the new attribute values, you reboot your system. The following subsystems provided by Digital support this type of boot-time modification:
  - `dli` – Data link interface subsystem
  - `generic` – Generic kernel subsystem
  - `ipc` – Interprocess communication subsystem
  - `lsm` – Logical Storage Manager
  - `net` – Network subsystem
  - `presto` – Prestoserve subsystem
  - `proc` – Process subsystem
  - `pts` – Pseudoterminal subsystem

- `rt` – Realtime subsystem
- `snmpinfo` – snmpinfo subsystem
- `streams` – STREAMS subsystem
- `tty` – Terminal subsystem
- `ufs` – UNIX File System
- `vfs` – System V File System
- `vm` – Virtual memory subsystem
- `xpr` – XPR kernel tracing subsystem

  If you decide to configure attributes of these subsystems, use the procedures described in Section 5.3.8.

If you purchase a device driver or other kernel subsystem from a company other than Digital, that product might also be dynamically loadable or allow you to dynamically configure attribute values. For information about dynamically configuring your kernel when working with products from other vendors, see the documentation for that product and Section 5.3.

If the subsystem you want to add, remove, or configure does not support dynamic configuration, you must use the static configuration method. You must also use this method to configure system parameters that do not support dynamic configuration. For information about the static configuration method, see Section 5.4.

5.3 Dynamic System Configuration

When you need to load, unload, or modify a dynamic subsystem, you use the `/sbin/sysconfig` command. This command has the following syntax:

/sbin/sysconfig [-h *hostname*] [-i *index* [-v|-c|-m|-q|-Q|-r|-s|-u]] [*subsystem-name*] [*attribute-list*]

You must be the superuser to load and unload subsystems.

You must also know the name of the subsystem you want to manage. You can determine the name of a subsystem by looking in the documentation that accompanies the subsystem or in the directories into which the subsystem is installed. Subsystems are installed in either the `/subsys` directory or the `/var/subsys` directory. When a subsystem is installed, a file named *`subsystem-name`*`.mod` appears in one of those two directories. You use that subsystem name as input to the `/sbin/sysconfig` command. The sections that follow describe the commands you use to manage subsystems.

You can load and unload subsystems on a local system or a remote system. For information about adding and removing subsystems on remote systems, see Section 5.3.7

If you are writing a loadable device driver or other loadable subsystem, refer to the *Writing Device Drivers: Tutorial* manual and the *Programmer's Guide*. The *Writing Device Drivers: Tutorial* manual describes the tasks performed by the system when you install a loadable device driver. This manual also describes how to write and package loadable device drivers. The *Programmer's Guide* gives general information about creating subsystems that are dynamically configurable and discusses the framework that supports dynamic configuration of subsystems and attributes.

5.3.1 Configuring Subsystems

To configure (load) a subsystem, enter the `sysconfig` command using the –c flag. Use this command whether you are configuring a newly installed subsystem or one that was removed using the `/sbin/sysconfig -u` command. For example, to configure the DECnet network (`decnet`) subsystem, issue the following command:

```
# /sbin/sysconfig -c decnet
```

5.3.2 Querying Subsystem State

Subsystems can be known to the kernel, but not available for use. To determine which subsystems are available for use, use the `/sbin/sysconfig –s` command. This command displays the state of all subsystems. Subsystems can have the following states:

- Loaded and configured (available for use)
- Loaded and unconfigured (not available for use but still loaded)

  This state applies only to static subsystems, which can be unconfigured, but cannot be unloaded.
- Unloaded (not available for use)

  This state applies only to loadable subsystems, which are automatically unloaded when you unconfigure them.

If you use the `/etc/sysconfig –s` command without specifying a subsystem name, a list of all the configured subsystems is displayed. For example:

```
# /sbin/sysconfig -s
cm: loaded and configured
generic: loaded and configured
proc: loaded and configured
io: loaded and configured
vm: loaded and configured
vfs: loaded and configured
ufs: loaded and configured
```

```
ipc: loaded and configured
tty: loaded and configured
xpr: loaded and configured
rt: loaded and configured
net: loaded and configured
dli: loaded and configured
lat: loaded and configured
bufcall: loaded and configured
strstd: loaded and configured
streams: loaded and configured
kinfo: loaded and configured
timod: loaded and configured
tirdwr: loaded and configured
xtiso: loaded and configured
dlb: loaded and configured
ldtty: loaded and configured
pts: loaded and configured
bba: loaded and configured
sfbp: loaded and configured
```

This list includes both statically linked subsystems and dynamically loaded subsystems.

To get information about the state of a single subsystem, include the name of the subsystem on the command line:

```
# /sbin/sysconfig -s lsm
lsm: unloaded
```

5.3.3 Determining Subsystem Type

You can determine whether a subsystem is dynamically loadable or static by using the `/sbin/sysconfig –m` command, as shown:

```
# /sbin/sysconfig -m kinfo lat
kinfo: static
lat: dynamic
```

The output from this command indicates that the `kinfo` subsystem is static, meaning that you must rebuild the kernel to add or remove that subsystem from the kernel. The `cmftest21` subsystem is dynamic, meaning that you can use the `sysconfig -c` command to configure the subsystem and the `sysconfig -u` command to unconfigure it.

5.3.4 Unloading a Subsystem

To unconfigure (and possibly unload) a subsystem, use the `/sbin/sysconfig -u` command, as shown:

```
# /sbin/sysconfig -u decnet
```

If you frequently configure and unconfigure device drivers you might notice that the device special files associated with a particular device driver differ from time to time. This behavior is normal. When you configure a device driver using the `/sbin/sysconfig` command, the system creates device special files. If you unload that device driver and load another one that uses the same `cdev` or `bdev` major numbers, the system removes the device special files for the unloaded device driver. Therefore, it must create new device special files the next time you configure the device.

5.3.5 Maintaining the List of Automatically Configured Subsystems

The system determines which subsystems to configure into the kernel at system reboot time by checking the list of automatically configured subsystems. The system configures each subsystem on that list, using the `sysconfig –c` command at each system reboot.

You maintain the list of automatically configured subsystems by using the `/sbin/init.d/autosysconfig` command.

This command has the following syntax:

/sbin/init.d/autosysconfig list [add *subsystem-name*] [delete *subsystem-name*]

Use the `/sbin/init.d/autosysconfig list` command to see a list of the loadable subsystems that the system automatically configures at each reboot.

To add a subsystem to the list, use the `/sbin/init.d/autosysconfig add` command. For example to add the `lat` subsystem, issue the following command:

```
# /sbin/init.d/autosysconfig add lat
```

If you unload a subsystem that is on the automatically configured subsystem list, you should remove that subsystem from the list. Otherwise, the system will configure the subsystem back into the kernel at the next system reboot. To remove the subsystem from the automatically configured subsystems list, issue the `/sbin/init.d/autosysconfig delete` command. For example, to delete the `lat` subsystem, issue the following command:

```
# /sbin/init.d/autosysconfig delete lat
```

5.3.6 Managing Subsystem Attributes

Occasionally, to improve the performance of a subsystem or of the system as a whole, you might modify the value of subsystem attributes. You use

the /sbin/sysconfig command to determine the names and values of subsystem attributes. You can also use the command to modify the value of a small number of attributes in the currently running kernel.

If you modify an attribute at run time by using the /sbin/sysconfig command, the modification persists as long as the system is running. If you shut down and reboot the system, the modification is lost. To modify subsystem attributes so that changes persist across reboots, store the attribute's value in the /etc/sysconfigtab database, as described in Section 5.3.8.

The system parameters that are set in the system configuration file and in the param.c file continue to define the system tables, and should be viewed as establishing default values in the kernel. You can override these values by using the /sbin/sysconfig command or by storing a value in the /etc/sysconfigtab database. For more information about the configuration file and param.c, see Section 5.4.

You can manage dynamic subsystem attributes either locally or remotely. For information on how to use the /sbin/sysconfig command remotely, see Section 5.3.7.

5.3.6.1 Determining the Value of Subsystem Attributes

Use the /sbin/sysconfig -q command to determine the value assigned to subsystem attributes. When you issue the /sbin/sysconfig -q command the subsystem you specify on the command line must be loaded and configured. For information about getting a list of the loaded and configured subsystems, see Section 5.3.2.

The following example shows how to use this command to determine which attributes are part of the generic subsystem:

```
# /sbin/sysconfig -q generic
generic:
clock-frequency = 1024
booted_kernel = vmunix
booted_args = vmunix modules=0xfffffc00005ea000
lockmode = 0
lockdebug = 0
locktimeout = 15
max-lock-per-thread = 8
lockmaxcycles = 0
rt_preempt_opt = 0
rt-preempt-opt = 0
cpu_enable_mask = 18446744073709551615
cpu-enable-mask = 18446744073709551615
msgbuf_size = 4096
```

```
message-buffer-size = 4096
dump-sp-threshold = 4096
lite-system = 0
```

The /sbin/sysconfig –q command lists all subsystem attributes and their values. Some attributes are configurable with the /sbin/sysconfig –r command. For information about which attributes are configurable, see *System Tuning and Performance Management*.

5.3.6.2 Identifying Dynamic Subsystem Attributes

You can identify which of a subsystem's attributes are dynamic by using the /sbin/sysconfig –Q command:

```
# /sbin/sysconfig -Q max-vnodes
vfs:
max-vnodes -    type=INT op=CRQ min_val=0 max_val=2147483647
```

This example shows using the –Q flag to get information about the max-vnodes attribute of the vfs subsystem. The max-vnodes attribute has the integer datatype, a minimum value of zero (0), and a maximum value of 2147483647. The op field indicates the operations that can be performed on the max-vnodes attribute. The following list describes the values that can appear in this field:

- C – The attribute can be modified when the subsystem is initially loaded.
- R – The attribute can be modified while the subsystem is running.
- Q – The attribute can be queried.

5.3.6.3 Modifying Dynamic Subsystem Attributes at Run Time

You can modify the value of an attribute at run time by issuing the /sbin/sysconfig –r command. The modification you make with this command persists until the next time the system is rebooted. When the system reboots, any changes made with the /sbin/sysconfig –r command are lost because the new value is not stored. The –r flag to the /sbin/sysconfig command is useful for testing a new subsystem attribute value. If the new value causes the system to perform as expected, you can later store it in the subsystem attribute database as described in Section 5.3.8.

When you use the /sbin/sysconfig –r command you specify the attribute, its new value, and the subsystem name on the command line. For example, to modify the dump-sp-threshold attribute for the generic subsystem, issue a command like the following:

```
# /sbin/sysconfig -r generic dump-sp-threshold=20480
```

To modify the value of more than one attribute at a time, include a list on the /sbin/sysconfig command line. For example, to modify the dump-sp-threshold attribute and the locktimeout attribute, issue a command like the following:

```
# /sbin/sysconfig -r generic dump-sp-threshold=20480
locktimeout=20
```

You do not include a comma between the two attribute specifications.

5.3.7 Managing Subsystems and Attributes Remotely

You can use the /sbin/sysconfig –h command to administer configurable subsystems and dynamic subsystem attributes remotely on a local area network (LAN). This ability allows you to administer several systems from a single machine.

Each system you want to administer remotely must have an /etc/cfgmgr.auth file that contains the full domain name of the local system. The name in the /etc/cfgmgr.auth file should be identical to the name in either the /etc/hosts file or in the Berkeley Internet Domain (BIND) or Network Information Service (NIS) hosts databases, if you are using BIND or NIS. You must create the /etc/cfgmgr.auth file; it is not on your system by default. The following shows an example cfgmgr.auth file:

```
salmon.zk3.dec.com
trout.zk3.dec.com
bluefish.zk3.dec.com
```

To manage subsystems and attributes on remote systems, you include the –h flag and a hostname with the /sbin/sysconfig command. For example, to load the decnet subsystem on a remote host named MYSYS, you issue the following command:

```
# /sbin/sysconfig -h MYSYS -c decnet
```

In the previous example, a decnet.mod file must exist in either the /subsys directory or the /var/subsys directory on the remote system before the subsystem can be loaded. If the loadable subsystem subset is kitted correctly, the *subsystem-name*.mod file is installed on the remote system when you use the setld command to install the loadable subsystem.

5.3.8 Managing the Subsystem Attributes Database

Information about dynamically configurable subsystem attributes is stored in the /etc/sysconfigtab database. You use this database to record the

values you want to be assigned to subsystem attributes each time the system is rebooted or a subsystem is configured. No attributes are set automatically in this database. If you want to change the default values of any attributes, you must include the subsystem name, the attribute name, and the value in the database yourself. You must be the superuser to modify the /etc/sysconfigtab database.

Note

The /etc/sysconfigtab database might contain stanza entries from a configurable subsystem's stanza.loadable file. This file and the entry in the /etc/sysconfigtab database are created automatically when you install certain configurable subsystems. You should not modify these entries in the database.

To add, update, or remove entries in the database, you create a stanza-format file containing names and values for attributes you want to modify. (For information about stanza-format files, see stanza(4)). For example, suppose you want to set the lockmode attribute in the generic subsystem to 1. To set this attribute, create a file named, for example, generic_attrs that has the following contents:

```
generic:
        lockmode = 1
```

After you create the stanza-format file, you use the /sbin/sysconfigdb command to update the /etc/sysconfigtab database. You name the stanza-format file on the command line using the –f flag. The sysconfigdb command reads the specified file and updates both the on-disk and in-memory copy of the database. However, the running kernel is not updated. (Use the sysconfig –r command to update the running kernel, as described in Section 5.3.6.3.)

The sysconfigdb command has the following syntax:

/sbin/sysconfigdb [-a|-d|-l|-m|-r|-s|-u] [-f *file*] [*subsystem-name*]

The sections that follow explain how to use the /sbin/sysconfigdb command to manage entries in the /etc/sysconfigtab database.

You can also use a text editor to add, update, or delete subsystem attributes in the /etc/sysconfigtab database. However, if you edit the /etc/sysconfigtab database, you must run the /sbin/sysconfigdb –s command after you write and quit the file so that the in-memory copy of the database is updated.

5.3.8.1 Listing Attributes in the Database

To list the entries in the /etc/sysconfigtab database, use the /sbin/sysconfigdb –l command. If you specify a subsystem name on the command line, the attributes of that subsystem are listed. Otherwise, all attributes defined in the database are listed.

For example, to list the attribute settings for the generic subsystem, issue the following command:

```
# /sbin/sysconfigdb -l generic
generic:
        lockmode = 0
```

5.3.8.2 Adding Attributes to the Database

To add subsystem attributes to the /etc/sysconfigtab database, enter the sysconfigdb –a command.

For example, to add the entries stored in a file named add_attrs to the database, issue the following command:

```
# /sbin/sysconfigdb -a -f add_attrs generic
```

5.3.8.3 Merging New Definitions into Existing Database Entries

To merge new definitions for attributes into an existing entry in the /etc/sysconfigtab database, enter the sysconfigdb –m command.

The sysconfigdb command merges the new definitions into the existing database entry as follows:

- If an attribute name does not appear in the database, the definition for that attribute is added to the database.
- If an attribute name does appear, the attribute receives the value specified by the new definition.
- If an attribute appears in the database, but is not included among the new definitions, its definition is maintained in the database.

For example, suppose that the following entry for the generic subsystem exists in the /etc/sysconfigtab database:

```
generic:
   lockmode = 4
   dump-sp-threshold = 6000
```

Suppose that you create a file named merge_attrs for updating this entry, which contains the following information:

```
generic:
  lockmode = 0
  lockmaxcycles = 4294967295
```

To merge the information in the `merge_attrs` file into the `/etc/sysconfigtab` database, issue the following command:

```
# /sbin/sysconfigdb -m -f merge_attrs generic
```

After the command finishes, the entry for the `generic` subsystem in the database appears as follows:

```
generic:
   lockmode = 0
   lockmaxcycles = 4294967295
   dump-sp-threshold = 6000
```

You can merge definitions for more than one subsystem into the `/etc/sysconfigtab` database with a single `sysconfigdb -m` command. For example, the `merge_attrs` file could contain new definitions for attributes in the `lsm` and `generic` subsystems. If you include more than one subsystem name in the `merge_attrs` file, you omit the subsystem name from the command line, as shown:

```
# /sbin/sysconfigdb -m -f merge_attrs
```

5.3.8.4 Updating Attributes in the Database

To update the entire definition of a subsystem that is already in the `/etc/sysconfigtab` database, enter the `/sbin/sysconfigdb –u` command.

For example, suppose the `generic` subsystem is defined as follows in the `/etc/sysconfigtab` file:

```
generic:
   lockmode = 4
   dump-sp-threshold = 6000
```

Suppose that you create a file named `update_attrs` for updating this entry, which contains the following information:

```
generic:
  lockmode = 0
  lockmaxcycles = 4294967295
```

To update the attributes, you issue the `sysconfigdb` command, as follows:

```
# /sbin/sysconfigdb -u -f update_attrs generic
```

After the command finishes, the entry for the `generic` subsystem in the database appears as follows:

```
generic:
   lockmode = 0
   lockmaxcycles = 4294967295
```

5.3.8.5 Removing Attribute Definitions from the Database

To remove the definitions of selected attributes from the /etc/sysconfigtab database, enter the /sbin/sysconfigdb –r command. The –r flag specifies that you want to remove the attribute definitions stored in a file from the database.

For example, suppose the generic subsystem is defined as follows in the /etc/sysconfigtab database:

```
generic:
  lockmode = 4
  dump-sp-threshold = 6000
```

To remove the definition of the dump-sp-threshold attribute, first create a file named remove_attrs that contains the following information:

```
generic:
   dump-sp-threshold = 6000
```

Then, issue the following command:

```
# /sbin/sysconfigdb -r -f remove_attrs generic
```

After the command finishes, the entry for the generic subsystem in the database appears as follows:

```
generic:
   lockmode = 4
```

The /sbin/sysconfigdb command removes only identical entries. In other words, the entries must have the same attribute name and value to be removed.

You can remove definitions of more than one attribute and for attributes in more than one subsystem from /etc/sysconfigtab database with a single sysconfigdb -r command. For example, the remove_attrs file could contain attribute definitions that you want to remove for the lsm and generic subsystems. If you include more than one subsystem in the remove_attrs file, you omit the subsystem name from the command line, as shown:

```
# /sbin/sysconfigdb -r -f remove_attrs
```

5.3.8.6 Deleting Subsystem Entries from the Database

To delete the definition of a subsystem from the /etc/sysconfigtab database enter the /sbin/sysconfigdb –d command.

For example, to delete the generic subsystem entry in the database, issue the following command:

```
# /sbin/sysconfigdb -d generic
```

The generic subsystem receives its default values the next time it is configured.

5.4 Static System Configuration

Static system configuration refers to the commands and files used to build and boot a new kernel and its static subsystems. The subsystems are viewed as static because they are linked directly into the kernel at build time. The steps you take to build a statically linked kernel vary depending upon why you want to modify the kernel.

If you modify the kernel to add a device driver, from Digital or from a company other than Digital, you follow these general steps:

- Install the device driver.
- If necessary, edit the target configuration file.

  In some cases, the device driver provides a Subset Control Program (SCP) that executes during the installation procedure and registers the driver in the necessary system configuration files. In this case, you need not edit the target configuration file yourself.

  If the device driver does not provide an SCP, you must edit the target configuration file yourself.
- Build a new kernel.

  If your device driver includes an SCP, build a new kernel by running the /usr/sbin/doconfig program as described in Section 5.4.3. If you need to edit the target configuration file before you build a new kernel, refer to Section 5.4.1.
- Shut down and reboot your system.

If you modify the kernel to add support for certain kernel options, you can build the new kernel by running the /usr/sbin/doconfig program and choosing the kernel option from a menu displayed during processing. You then shutdown and reboot your system.

To determine which kernel options you can configure in this way, issue the /usr/sbin/kopt command. The command displays a list of kernel options and prompts you for kernel options selections. To exit from the /usr/sbin/kopt command without choosing options, press the Return key. For information about running the /usr/sbin/doconfig program to add kernel options using a menu, see Section 5.4.2.

If you build a new static kernel for any other reason, you must modify one or more system files as part of rebuilding the kernel. The system files you modify depend upon the change you want to make to the kernel:

- You modify the target configuration file to make changes to keywords that, for example, define the kernel you want to build, define devices, or define pseudodevices. You can also edit this file to change the value of system parameters. For details about the contents of the target configuration file, see Section 5.5.
- You remove certain static subsystems from the kernel by removing (or commenting out) their entry from a file in the `/usr/sys/conf` directory. For information about this file, see Section 5.5.2.
- You modify the `param.c` file to change the value of system parameters. You modify these parameters to tune your system's performance. For information about the `param.c` file, see Section 5.5.3.

Normally, you make these changes using the text editor of your choice before you begin building the kernel. (Alternatively, you can edit the system configuration file during the kernel building procedure. However, if you choose to edit the configuration file during the procedure, define the `EDITOR` environment variable to be the editor of your choice. The default editor is the `ed` line editor.) For information about running the `/usr/sbin/doconfig` program to build a kernel after editing system files, see Section 5.4.3.

5.4.1 Building the Kernel to Add Support for a New Device

When you add a new device to the system and the device installation includes no SCP, you must edit the target configuration file to allow the operating system to support the new device. You include the device definition keyword in the target configuration file. Because Digital UNIX supports many devices, determining which keyword to add to your target configuration file can be difficult.

The following procedure explains how to determine which device definition keyword to add to your target configuration file and how to rebuild the kernel once you have edited the target configuration file. The procedure assumes that you do not know the appropriate keyword to add. In some cases, you might be able to determine the appropriate keyword by looking at documentation supplied with the hardware or with a new version of Digital UNIX. Another source of this information is an existing configuration file on another system that already has the device connected to it. If you know what keyword you need to add to your system, use a text editor to add that keyword to your target configuration file and rebuild the kernel as described in Section 5.4.3.

If you are unsure of the keyword you need to add to the target configuration file for your system, connect the new device to the system as directed in the hardware manual and use the following procedure:

1. Log in as root or become the superuser and set your default directory to the /usr/sys/conf directory.
2. Save a copy of the existing /vmunix file. If possible, save the file in the root (/) directory, as follows:

   ```
   # cp /vmunix /vmunix.save
   ```

   If there are disk space constraints, you can save the kernel file in a file system other than root. For example:

   ```
   # cp /vmunix /usr/vmunix.save
   ```

3. Shutdown and halt the system as follows:

   ```
   # shutdown -h now
   ```

4. At the console prompt, boot the generic kernel, /genvmunix. The generic kernel contains support for all valid devices, so if you boot it during the process of adding a new device to your target kernel, the new device is known to the kernel. To boot the generic kernel, issue the following command:

   ```
   >>> boot -fi "genvmunix"
   ```

Note

If the /genvmunix file does not exist on your system, or the generic kernel fails to recognize the device you are adding, rebuild the generic kernel.

To rebuild the generic kernel, you must have installed all the required and optional kernel subsets. You can get a list of the kernel subsets, including information about whether or not they are installed, by issuing the following command:

```
# /usr/sbin/setld -i | grep Kernel
```

After all kernel subsets are installed, issue the following command:

```
# doconfig -c GENERIC
```

The –c flag specifies that you want to build a kernel using an existing configuration file, in this case the GENERIC configuration file. For more information about building a kernel from an existing configuration file, see Section 5.4.3.

After the generic kernel is running and recognizes the new device, continue with step 5. When the build ends, condider using the `strip` command to reduce the size of the kernel. See the `strip`(1) reference page.

---

5. At the single-user mode prompt, check and mount local file systems by issuing the following command, unless you are using the Logical Storage Manager software (LSM):

   ```
   # /sbin/bcheckrc
   ```

   If you are using the Logical Storage Manager (LSM) software, check local file systems and start LSM by issuing the following command:

   ```
   # /sbin/lsmbstartup
   ```

6. Run the `sizer` program to size your system hardware and create a new target configuration file that includes the new device:

   ```
   # sizer -n MYSYS
   ```

   The `sizer -n` command creates a new target configuration file for your system that includes the appropriate device definition keyword for the new deivce. (This process is similar to the process that occurs at system installation time. For more information, see Section 5.1.) The `sizer` program stores the new target configuration file in the `/tmp` directory.

7. Compare the new target configuration file created by `sizer` with the existing target configuration file for your system:

   ```
   # diff /tmp/MYSYS MYSYS
   ```

   Check the differences between these files until you find the new device definition keyword. (The two files might differ in other ways if you have customized your existing configuration file, such as by specifying a nondefault value for the `maxusers` option.)

8. Use the text editor of your choice to add the new device definition keyword to your existing configuraton file (in this case, MYSYS). Adding the new keyword allows your existing configuration file to support the new device, without losing any changes you made to that file in the past.

Note

If you add or remove communications devices from your configuration file, you must edit the `/etc/inittab` file and the `/etc/securettys` file to match your new configuration;

that is, to match the /dev/tty*nn* special device files. For more information, see inittab(4) and securettys(4).

---

9. Build a new kernel by issuing the following /usr/sbin/doconfig command:

```
# /usr/sbin/doconfig -c MYSYS

*** KERNEL CONFIGURATION AND BUILD PROCEDURE ***

Saving /usr/sys/conf/MYSYS as /usr/sys/conf/MYSYS.bck
```

   Answer the following prompt to indicate that you do not want to edit the configuraton file:

```
Do you want to edit the configuration file? (y/n) [n]: n

*** PERFORMING KERNEL BUILD ***
⋮
The new kernel is /usr/sys/MYSYS/vmunix
```

10. When the kernel configuration and build process completes without errors, move the new vmunix file to /vmunix. On a system named MYSYS, issue the following command:

```
# mv /usr/sys/MYSYS/vmunix /vmunix
```

11. Reboot the system as follows:

```
# /usr/sbin/shutdown -r now
```

If the new /vmunix file fails to boot, boot using the kernel you saved at the beginning of the procedure. To use the saved kernel, follow these steps:

1. Check all local file systems by using the fsck –p command as follows:

```
# fsck -p
```

2. Write-enable the root file system by using the mount –u command as follows:

```
# mount -u /
```

3. If necessary, mount the file system where the /vmunix.save file is stored. For example, if you copied the /vmunix file to the /usr file system, issue the following command:

```
# mount /usr
```

4. Restore the saved copy. For example, if you saved your running kernel in the /vmunix.save file, issue the following command:

   # **mv /vmunix.save /vmunix**

5. Shutdown and reboot the system, as follows:

   # **shutdown -r now**

After your system is running again, you can modify the target configuration file as needed and rebuild the kernel starting at step 3.

5.4.2 Building the Kernel to Add Selected Kernel Options

If you invoke the /usr/sbin/doconfig program without using flags, you are given the opportunity to modify the kernel using a menu. To modify the kernel using a menu, follow these steps:

1. Log in as root or become the superuser and set your default directory to the /usr/sys/conf directory.
2. Save a copy of the existing /vmunix file. If possible, save the file in the root (/) directory, as follows:

   # **cp /vmunix /vmunix.save**

   If there are disk space constraints, you can save the kernel file in a file system other than root. For example:

   # **cp /vmunix /usr/vmunix.save**

3. Run the /usr/sbin/doconfig program using no flags, as follows:

   # **/usr/sbin/doconfig**

   ```
   *** KERNEL CONFIGURATION AND BUILD PROCEDURE ***

   Saving /usr/sys/conf/MYSYS as /usr/sys/conf/MYSYS.bck
   ```

4. Enter the name of the configuration file at the following prompt:

   ```
   Enter a name for the kernel configuration file. [MYSYS]: MYSYS
   ```

   The kernel configuration processes convert the system name to uppercase when determining what name to supply as the default configuration file name. For example, on a system named mysys, the default configuration file is named MYSYS.

   If the configuration file name you specify does not currently exist, the /usr/sbin/doconfig program builds one with that name. Continue this process by selecting the kernel options in step 10.

5. If the configuration file name you specify exists, answer the following prompt to indicate that you want to overwrite it:

```
A configuration file with the name MYSYS already exists.
Do you want to replace it? (y/n) [n]: y
```

6. Select kernel options from a menu similar to the following one:

```
*** KERNEL OPTION SELECTION ***

 Selection   Kernel Option
-------------------------------------------------------------
-
     1       System V Devices
     2       NTP V3 Kernel Phase Lock Loop (NTP_TIME)
     3       Kernel Breakpoint Debugger (KDEBUG)
     4       Packetfilter driver (PACKETFILTER)
     5       Point-to-Point Protocol (PPP)
     6       STREAMS pckt module (PCKT)
     7       X/Open Transport Interface (XTISO, TIMOD, TIRDWR)
     8       File on File File System (FFM)
     9       ISO 9660 Compact Disc File System (CDFS)
     10      Audit Subsystem
     11      ACL Subsystem
     12      LAN Emulation over ATM (LANE)
     13      Classical IP over ATM (ATMIP)
     14      ATM UNI 3.0/3.1 Signalling for SVCs
     15      Asynchronous Transfer Mode (ATM)
     16      Advanced File System (ADVFS)
     17      All of the above
     18      None of the above
     19      Help
-------------------------------------------------------------
-

Enter the selection number for each kernel option you want.
For example, 1 3 [18]:
```

7. Answer the following prompt to indicate whether or not you want to edit the configuration file:

```
Do you want to edit the configuration file? (y/n) [n]:
```

   You need not edit the configuraton file unless you have changes other than adding one or more of the subsystems in the menu to the kernel.

   If you choose to edit the configuration file, the `/usr/sbin/doconfig` program invokes the editor specified by the `EDITOR` environment variable.

   For information about the configuration file, see Section 5.5

   After you finish editing the configuration file, the `/usr/sbin/doconfig` program builds a new kernel.

8. When the kernel configuration and build process completes without errors, move the new vmunix file to /vmunix. On a system named MYSYS, issue the following command:

   # **mv /usr/sys/MYSYS/vmunix /vmunix**

9. Reboot the system as follows:

   # **/usr/sbin/shutdown -r now**

If the new /vmunix file fails to boot, boot using the kernel you saved at the beginning of the procedure. To use the saved kernel, follow these steps:

1. Check all local file systems by using the fsck –p command as follows:

   # **fsck -p**

2. Write-enable the root file system using the mount –u command as follows:

   # **mount -u /**

3. If necessary, mount the file system where the /vmunix.save file is stored. For example, if you copied the /vmunix file to the /usr file system, issue the following command:

   # **mount /usr**

4. Restore the saved copy. For example, if you saved your running kernel in the /vmunix.save file, issue the following command:

   # **mv /vmunix.save /vmunix**

5. Shutdown and reboot the system, as follows:

   # **shutdown -r now**

After your system is running again, you can modify the target configuration file as needed and rebuild the kernel starting at step 3.

5.4.3 Building a Kernel After Editing System Files

If you or an SCP modify system files, such as the target configuration file, you can rebuild your kernel using the /usr/sbin/doconfig -c command. The –c flag allows you to name an existing configuration file, which the /usr/sbin/doconfig program uses to build the kernel. To build a new kernel using an existing configuration file, follow these steps:

1. Log in as root or become the superuser and set your default directory to the /usr/sys/conf directory.
2. Save a copy of the existing /vmunix file. If possible, save the file in the root (/) directory, as follows:

```
# cp /vmunix /vmunix.save
```

If there are disk space constraints, you can save the kernel file in a file system other than root. For example:

```
# cp /vmunix /usr/vmunix.save
```

3. Run the /usr/sbin/doconfig program specifying the name of the target configuration file with the –c flag. For example on a system named MYSYS, enter the following command:

```
# /usr/sbin/doconfig -c MYSYS

*** KERNEL CONFIGURATION AND BUILD PROCEDURE ***

Saving /usr/sys/conf/MYSYS as /usr/sys/conf/MYSYS.bck
```

4. Answer the following prompt to indicate whether or not you want to edit the configuration file:

```
Do you want to edit the configuration file? (y/n) [n]:
```

If you modified the configuration file before you started this procedure, indicate that you do not want to edit the configuration file.

If you choose to edit the configuration file, the /usr/sbin/doconfig program invokes the editor specified by the EDITOR environment variable.

For information about the configuration file, see Section 5.5

After you finish editing the configuration file, the /usr/sbin/doconfig program builds a new kernel.

5. When the kernel configuration and build completes without errors, move the new vmunix file to /vmunix. On a system named MYSYS, issue the following command:

```
# mv /usr/sys/MYSYS/vmunix /vmunix
```

6. Reboot the system as follows:

```
# /usr/sbin/shutdown -r now
```

If the new /vmunix file fails to boot, boot using the kernel you saved at the beginning of the procedure. To use the saved kernel, follow these steps:

1. Check all local file systems by using the fsck –p command as follows:

```
# fsck -p
```

2. Write-enable the root file system using the mount –u command as follows:

```
# mount -u /
```

3. If necessary, mount the file system where the /vmunix.save file is stored. For example, if you copied the /vmunix file to the /usr file system, issue the following command:

   # **mount /usr**

4. Restore the saved copy. For example, if you saved your running kernel in the /vmunix.save file, issue the following command:

   # **mv /vmunix.save /vmunix**

5. Shutdown and reboot the system, as follows:

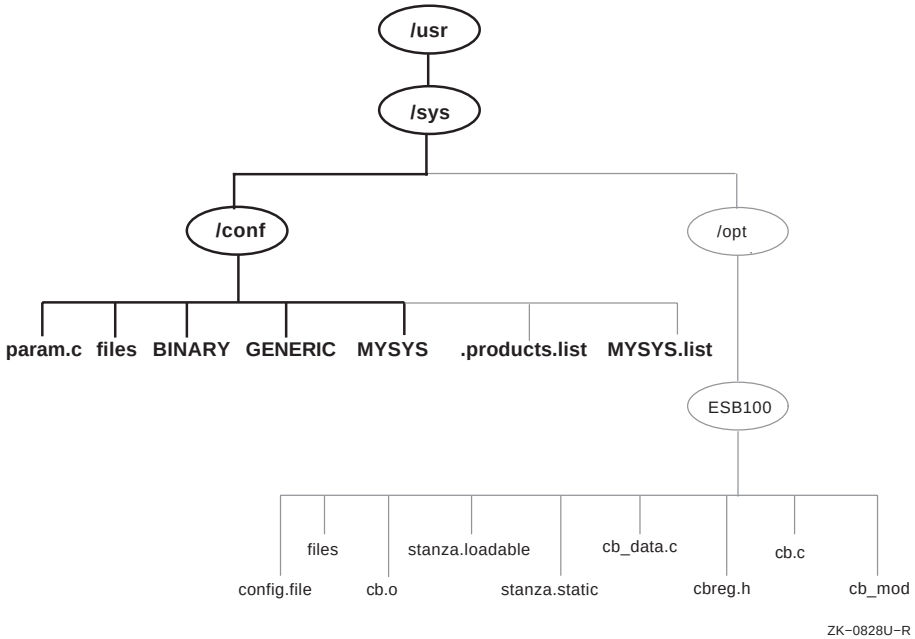
   # **shutdown -r now**

After your system is running again, you can modify the target configuration file as needed and rebuild the kernel starting at step 3.

5.5 Static Configuration Files

To build and run a working kernel, the system depends on the presence of specific directories under the /usr/sys directory. Figure 5–1 shows the directory structure of the system configuration files. The dotted lines indicate optional directories and files for third-party static subsystems.

Figure 5–1: Configuration Files Directory Hierarchy



As shown in Figure 5–1, the /usr/sys/conf directory contains files that define the kernel configuration for the generic and target kernels. These files represent the configuration of the static portion of the kernel. When you work with the system files to reconfigure the kernel, you are interested primarily in five files:

- /usr/sys/conf/*NAME*
- /usr/sys/conf/GENERIC
- /usr/sys/conf/.product.list
- /usr/sys/conf/*NAME* .list
- /usr/sys/conf/param.c

The sections that follow provide more information about these files.

5.5.1 System Configuration Files

The /usr/sys/conf directory contains two important system configuration files:

- The target configuration file, /usr/sys/conf/*NAME,* is a text file that defines the components that the system builds into your kernel. By convention, the *NAME* portion of the pathname is the name of your system in capital letters. For example, a system named MYSYS is described by a file named /usr/sys/conf/MYSYS. Each system has a target configuration file built for it by the sizer program during system installation. You modify the target configuration file when you want to change one of the following keyword definitions:
  - Global keywords that, if you are managing more than one system, are often defined the same across systems
  - System definition keywords that describe the kernel you want to build for a particular system
  - Device definition keywords that describe the devices connected to a particular system
  - callout keyword definitions that allow you to run shell command subprocesses during kernel configuration
  - options keyword definitions that specify software to be compiled into the system
  - makeoptions keyword definitions that are passed to the compiler, assembler, and linker when building the kernel
  - pseudodevice keyword definitions that describe psuedodevices used on the system

  Table 5–2 lists the entries that can be included in the target configuration file.

- The /usr/sys/conf/GENERIC configuration file is the configuration file that describes the generic kernel. The generic kernel supports all valid devices and is useful when you are adding a new device to the system. You can also use the generic kernel as a back-up kernel should your target kernel be corrupted in some way.

  Avoid deleting the /genvmunix file, which contains the generic kernel. If you accidentally delete the generic kernel, you can rebuild it by using the doconfig -c GENERIC command. For more information about building a kernel using an existing configuration file, see Section 5.4.3.

Note

Never delete the /usr/sys/conf/GENERIC file.

5.5.2 Extensions to the Target Configuration File

The /usr/sys/conf directory contains two optional configuration files that describe extensions to the target configuration file. These are the /usr/sys/conf/.product.list file and the /usr/sys/conf/*NAME* file. These files store information about static kernel subsystems, sometimes called kernel layered products.

When you install a static subsystem, its SCP normally edits the /usr/sys/conf/.product.list file and adds an entry for the subsystem. After the SCP completes, you run the /usr/sbin/doconfig program to configure the new subsystem into the kernel.

The /usr/sbin/doconfig program creates the *NAME*.list file. The *NAME* variable is the same as the target configuration file, and by convention, is your system name in capital letters. For example, the *NAME*.list file for a system named MYSYS is MYSYS.list.

If you need to modify your system because of a third-party layered product (for example, to remove a layered product from the kernel being built), make the necessary modifications to the *NAME*.list file and build a new kernel.

Each entry in the *NAME*.list file consists of six fields separated by a colon (:). The following example is part of a *NAME*.list file and shows an entry for a static kernel subsystem that has been loaded into the /usr/sys/opt/ESB100 directory:

```
/usr/sys/opt/ESB100:UNXDASH100:920310100739:DASH Systems:controlsys:100
     1          2       3        4       5     6
```

The fields in this entry contain the following information:

1. The full pathname where the system configuration tools will find extensions to input data. This location can contain files such as:
   - Product-specific configuration files
   - The `config.file` file fragment (contains keywords related only to the product)
   - The `files` file fragment (contains information about the location of the product's source code, when the product should be loaded into the kernel, and whether source or binary code is provided)
   - The `stanza.static` file (contains information about a static driver's major number requirements and the names and minor numbers of the device special files)
   - Object files
   - Source code files
2. The `setld` subset identifier.
3. The date and time that the product is ready for distribution.
4. The name of the company that provided the subsystem.
5. The product name.
6. The `setld` 3-digit product version code.

The order of the line entries in the *NAME*`.list` file reflects the order in which the entries are processed.

The `/usr/sbin/doconfig` program creates the *NAME*`.list` file by copying the `.product.list` file, if it exists. Note that when using the `/usr/sbin/doconfig -c` command, `/usr/sbin/doconfig` uses the existing *NAME*`.list` file. If the `.product.list` file changes (for example, a new kernel layered product was installed) and the –`c` flag is used, either delete the *NAME*`.list` file or manually edit it before invoking `/usr/sbin/doconfig` to propagate the change in the `.product.list` file to the *NAME*`.list` file.

You can also create the file by copying the `.product.list` file to the *NAME*`.list` file. You can then edit the *NAME*`.list` file and either delete the lines that you do not want built into the kernel or comment them out by putting a number sign (#) as the first character in each line that you do not want.

Note

Never edit the `.product.list` file.

Refer to the *Writing Device Drivers: Tutorial* manual for more information on the *NAME*.list and .product.list files.

5.5.3 The param.c File

The param.c file contains default values for a number of system parameters. You use these parameters to tune your system's performance. Table 5–1 lists system parameters that you can modify. For information about deciding what values to assign to these parameters, see *System Tuning and Performance Management*.

In some cases, as noted in Table 5–1, a parameter in the param.c file can also be included in your target configuration file. In this case, a value specified in the configuration file overrides the value specified in the param.c file. Therefore, if you modify the value of a system parameter in the param.c file, be sure to remove the corresponding entry from the target configuration file.

Table 5–1: Tunable param.c File Entries

Parameter	Default Value	Configuration File Equivalent
autonice	0	Set to 1 if AUTONICE defined
bufcache	3	bufcache
bufhsz	512	None
hz	100 CLOCKS_PER_SEC	None, defined in <machine/ mach time.h>
create_fastlinks	1	None
inohsz	512	None
maxuprc	64	maxuprc
maxuthreads	256	maxuthreads
maxusers	system dependent	maxusers
ncallout	36+8*maxuser+threadmax	maxcallouts
nchsize	NVNODE*11/10	None
nchsz	128	None
nclist	120 if clist ptys (pseudo-device pty #) are configured, the default value is 60+12*maxusers	nclist

Table 5–1: Tunable param.c File Entries (cont.)

Parameter	Default Value	Configuration File Equivalent
ndquot	NVNODE+(MAXUSERS * NMOUNT/4	None
nmount_max	256	None
nquota	(MAXUSERS*9)/7+3	None
nvnode	NPROC+(2*MAXUSERS)+128	None
open_max_hard	4096	None
open_max_soft	4096	None
path_num_max	64	None
port_hash_max_num	50*(6*threadmax+2000)	Based on threadmax
port_max_num	6*threadmax+2000	Based on threadmax
port_reserved_max_num	6*threadmax+2000	Based on threadmax
set_max_num	2*threadmax+200	Based on threadmax
select_max_elements	1024+NPROC*4	None
select_chunk_elements	256	None
spechsz	64	None
sys_v_mode	0	sys_v_mode
task_max	1 + (20 + (8*maxusers))	task_max
threadmax	8192	threadmax
ucred_max	128	None
ubc_minpercent	10	ubcminpercent
ubc_maxpercent	100	ubcmaxpercent
ufs_blkpref_lookbehind	8	None
vm_max_wrpgio_kluster	32*1024	writeio_kluster
vm_max_rdpgio_kluster	16*1024	readio_kluster

5.6 Configuration File Entries

The system configuration file contains the following keyword definitions:

- Global keyword definitions
- System definition keywords
- Device definition keywords
- `callout` keyword definitions
- `options` keyword definitions
- `makeoptions` keyword definitions
- `pseudo-device` keyword definitions

Table 5–2 lists the available configuration options. The first column specifies the configuration file keyword. The second column provides the default value assigned to the keyword if it is not included in the configuration file. The third column states whether you can change the value of the keyword. The fourth column states whether the keyword is required to be present in the configuration file.

The sections that follow the table define some of these entries.

Note

The configuration files supplied with Digital UNIX, the `GENERIC` file and the target configuration file for your system that is generated by the `sizer` program at installation time, override the default values for certain options. For example, the default value for the `maxdsiz` option is 32 MB; however, the configuration files supplied with Digital UNIX increase `maxdsiz` to 1 gigabyte.

Table 5–2: Configuration File Entries

	Default	Configurable	Required
Global Keywords:			
bufcache	3		
cpu	DEC2000_300		
	DEC2100_500		
	DEC3000_300		
	DEC3000_400		
	DEC3000_500		
	DEC4000		
	DEC7000		

Table 5–2: Configuration File Entries (cont.)

	Default	Configurable	Required
dfldsiz	134217728	Yes	No
dflssiz	1048576	Yes	No
heappercent	7	Yes	No
ident	GENERIC	No	Yes
kentry_zone_size	16777216	Yes	No
machine	alpha	Yes	Yes
mapentries	200	Yes	No
maxcallouts	system dependent	Yes	No
maxdsiz	1073741824	Yes	No
maxssiz	33554432	Yes	No
maxuprc	64	Yes	No
maxuthreads	256	Yes	No
maxusers	system dependent	Yes	Yes
maxvas	1073741824	Yes	No
max_vnodes	system dependent	Yes	No
msgmax	8192	Yes	No
msgmnb	16384	Yes	No
msgmni	50	Yes	No
msgtql	40	Yes	No
processors	1	No	Yes
scs_sysid	1	No	Yes
segmentation	1	Yes	No
semaem	16384	Yes	No
semmni	10	Yes	No
semmns	60	Yes	No
semmsl	25	Yes	No
semopm	10	Yes	No
semume	10	Yes	No
semvmx	32767	Yes	No
shmmin	1	Yes	No

Table 5–2: Configuration File Entries (cont.)

	Default	Configurable	Required
shmmax	4194304	Yes	No
shmmni	100	Yes	No
shmseg	32	Yes	No
swapbuffers	128	Yes	No
sys_v_mode	0	Yes	No
task_max	system dependent	Yes	No
threadmax	system dependent	Yes	No
timezone	0 dst 0	Yes	No
ubcbuffers	256	Yes	No
vpagemax	16384	Yes	No
zone_size	67108864	Yes	No
System Definition Keyword:			
config	vmunix swap generic	Yes	Yes
Device Definition Keywords:			
controller	All supported controller types	Yes	Yes
bus	All supported bus types	Yes	Yes
device disk	All supported disk device types	Yes	Yes
device tape	All supported tape device types	Yes	Yes
callout Keywords:			
at_start	none	Yes	No
at_exit	none	Yes	No
at_success	none	Yes	No
before_h	none	Yes	No
after_h	none	Yes	No
before_makefile	none	Yes	No
after_makefile	none	Yes	No
before_c	none	Yes	No

Table 5–2: Configuration File Entries (cont.)

	Default	Configurable	Required
after_c	none	Yes	No
before_conf	none	Yes	No
after_conf	none	Yes	No
options Keywords:			
AUTONICE	Off	Yes	No
BSD_TTY	On	No	Yes
BUFCACHE_STATS	On	No	Yes
BIN_COMPAT	On	No	Yes
CDFS	On	Yes	No
COMPAT_43	On	No	Yes
DLI	Off	Yes	No
DLB	On	Yes	No
FFM_FS	Off	Yes	No
GENERIC	On	No	Yes
INET	On	No	Yes
INOCACHE_STATS	On	No	Yes
KDEBUG	Off	Yes	No
LABELS	On	No	Yes
LAT	On	Yes	No
LDTTY	On[a]	Yes	No
LMF	On	No	Yes
MACH	On	No	Yes
MACH_COMPAT	On	No	Yes
MACH_CO_INFO	On	No	Yes
MACH_DEVICE	On	No	Yes
MACH_IPC_STATS	On	No	Yes
MACH_IPC_TCACHE	On	No	Yes
MACH_IPC_WWA	On	No	Yes
MACH_IPC_XXXHACK	On	No	Yes

Table 5–2: Configuration File Entries (cont.)

	Default	Configurable	Required
MACH_NET	On	No	Yes
MACH_SCTIMES	On	No	Yes
MAX_BDEVSW	70 (minimum is 50)	Yes	No
MAX_CDEVSW	125 (minimum is 75)	Yes	No
NFS	On	No	Yes
NFS_SERVER	On[a]	Yes	No
NTP_TIME	Off	Yes	No
OSF	On	No	Yes
OSF_MACH_O	Off	No	No
PACKETFILTER	Off	Yes	No
PCKT	Off	Yes	No
PROC_FS	On[a]	Yes	No
QUOTA	On[a]	No	Yes
RT[b]	On	No	Yes
RT_PML	Off	Yes	Yes
RT_PREEMPT	On	Yes	No
RT_PREEMPT_OPT	Off	Yes	Yes
RT_SCHED	On	No	Yes
RT_SCHED_RQ	On	No	Yes
RT_SEM	On	No	Yes
RT_TIMER	On	No	Yes
SER_COMPAT	On	No	Yes
SL	On[a]	Yes	No
SNMPINFO	On	No	Yes
STAT_TIME	On	No	Yes
STREAMS	On[a]	No	Yes
STRIFNET	Off	Yes	No
STRKINFO	On[a]	No	Yes
SYSV_COFF	On	No	Yes

Table 5–2: Configuration File Entries (cont.)

	Default	Configurable	Required
SYSV_ELF	On	No	Yes
SYSV_FS	Off	No	No
RPTY [c]	On[a]	Yes	No
TIMOD	Off	Yes	No
TIRDWR	Off	Yes	No
TRN	Off	Yes	No
TRSRCF	Off	Yes	No
UFS	On	No	Yes
UIPC	On	No	Yes
ULT_BIN_COMPAT	On	No	Yes
UNIX_LOCKS	On	No	No
VAGUE_STATS	On	No	Yes
XTISO	Off	Yes	No
makeoptions Keywords:			
ASOPTS	="-w" (Off)	Yes	No
CDEBUGOPTS	="-g3"	No	Yes
LDOPTS	="-x" (Off)	Yes	No
LOADADDR	="fffffc000023000"	Yes	No
Pseudodevice Keywords:			
cpus <16>	On, 1	No	Yes
ether	On	No	Yes
gwscreen	Off	Yes	No
loop	On	No	Yes
lv <num>	On, 3	Yes	Yes
lsm	On, 1	Yes	Yes
presto	Off	Yes	No
pty <num>[c]	Off	Yes	Yes
rt_hab	On	No	Yes
soe_two_hab	On	No	Yes

Table 5–2: Configuration File Entries (cont.)

	Default	Configurable	Required
strpush <num>	On, 16	Yes	Yes
svid_three_hab	On	No	Yes
svr_four_hab	On	No	Yes
sysv_hab	On	No	Yes
ws	Off (On for workstations)	Yes	No

[a]On systems with 24 MBs of memory, this option is not configured into the kernel by default.
[b]RT refers to Digital's Realtime software.
[c]The table indicates that the pty psuedodevice keyword is required and the RPTY option is not required. It does not matter which one is configured, as long as one is configured.

5.6.1 Global Keywords

Global keywords specify system-wide definitions. The following sections describe these keywords.

5.6.1.1 Kernel Identification

The `ident` keyword identifies the kernel that you are building. In general, you identify the kernel according to the machine it runs on; by convention, the kernel name is in uppercase letters. For example, the identification for a kernel that runs on a machine named MYSYS would have the following `/usr/sys/conf/MYSYS` configuration file entry:

```
ident   MYSYS
```

5.6.1.2 Time Zone

The Digital UNIX kernel does not store time zone information. The `timezone` keyword sets the initial value of the kernel's `tz` structure, which is used only for backward compatibility with executables that use the `gettimeofday` function. The `tz` structure maintains its initial value as long as the system is in single-user mode. The `tz` structure is overwritten by the local time zone when the system boots to multiuser mode.

By default, the `timezone` keyword is specified as follows:

```
timezone        0 dst 0
```

Refer to Section 4.4 for information about configuring the time zone.

5.6.1.3 Process Memory Size Limits

Some keywords define the default and maximum size limits for the data and stack segments in the address space of a process. The default size is the initial limit. The maximum size is the hard limit or the absolute limit. You can use the C shell `limit` command and the `getrlimit` and `setrlimit` system calls to change these limits. You can set these limits in the configuration file by using the following keywords:

Keyword	Usage
`dfldsiz`	Default data segment size limit
`maxdsiz`	Maximum data segment size limit
`dflssiz`	Default stack size limit
`maxssiz`	Maximum stack size limit

5.6.1.4 System V Functionality

The `sys_v_mode` keyword specifies whether the kernel exhibits System V behavior when it sets the group ID and file mode for newly created files. If the `sys_v_mode` keyword is set to 0 (zero), System V functionality is not enabled; this is the default. If the `sys_v_mode` keyword is set to 1, System V functionality is enabled.

This system keyword directly affects the `open`(), `creat`(), and `mkdir`( ) system calls.

The following tables describe how the `sys_v_mode` keyword affects behavior during file creation, directory creation, and file creation using the `open`( ) system call.

During file creation, the value of the group ID of any created file is affected regardless of how the `S_ISGID` bit is set initially. In the following table, the first column indicates that the System V keyword is enabled. The second column specifies how the `S_ISGID` bit is set in the parent directory. The third and fourth columns reflect how a created file is affected by the settings in columns 1 and 2.

Keyword	Parent Directory S_ISGID bit	New File Group ID	New File S_ISGID bit
`1`	Clear	Same as process GID	Clear
`1`	Set	Same as parent directory	Clear

For directory creation, both the value of the group ID of any created directory and the value of the S_ISGID bit are affected. In the following table, the first column indicates that the System V keyword is enabled. The second column specifies how the S_ISGID bit is set in the parent directory. The third and fourth columns reflect how a created directory is affected by the settings in columns 1 and 2.

Keyword	Parent Directory S_ISGID bit	New Directory Group ID	New Directory S_ISGID bit
1	Clear	Same as process GID	Clear
1	Set	Same as parent directory	Set

The next table shows how a created file is affected when the open() system call is used to forcibly set the S_ISGID bit. Note that the System V keyword is also enabled. Column 1 indicates the setting of the S_ISGID bit and columns 2 and 3 show how the created file is affected.

File Creation Mode S_ISGID bit	New File Group ID Equals Effective Group ID of Process or Support Group Member	New File S_ISGID bit
Clear	Yes	Clear
Clear	No	Clear
Set	Yes	Set
Set	No	Clear

If the keyword is not set as in the previous table, the S_ISGID bit is always cleared, as per the base operating system and the POSIX interface.

5.6.1.5 System V IPC

The following keywords define the System V IPC parameters (messages, semaphores, and shared memory):

Messages Keywords	Usage
msgmax	Maximum message size
msgmnb	Maximum number of bytes on queue
msgmni	Number of message queue identifiers
msgtql	Number of system message headers

Semaphores Keywords	Usage
semaem	Adjust on exit maximum value
semmni	Number of semaphore identifiers
semmns	Number of semaphores in the system
semmsl	Maximum number of semaphores per ID
semopm	Maximum number of semaphores per semop call
semume	Maximum number of undo entries per process
semvmx	Semaphore maximum value
Shared Memory Keywords	**Usage**
shmmax	Maximum shared memory segment size
shmmin	Minimum shared memory segment size
shmmni	Number of shared memory identifiers
shmseg	Maximum number of attached shared memory segments per process

5.6.1.6 Expected Number of Simultaneous Users

The maxusers keyword defines the number of simultaneous users that your system can support without straining system resources. The number should not be taken literally; from a performance standpoint, the number should always be greater than the expected number of real users. This number also is not the number of logins specified in your system software license. Refer to the Software Product Description (SPD) for information on the maximum number of supported users.

The default value assigned to maxusers depends upon the size of your system. For systems that have 24 MB of memory (small memory systems), the default value is 16. For all other systems, the default value is 32.

System algorithms use the maxusers keyword to size a number of system data structures and to determine the amount of space allocated to system tables. One such table is the system process table, which is used to determine how many active processes can be running at one time.

Increasing the value of maxusers allocates more system resources for use by the kernel. However, it also increases the amount of physical memory consumed by the kernel. Decreasing the value of maxusers reduces kernel memory usage, but also allocates less system resources. The setting of maxusers should be a balance between the number of users and the system hardware configuration (primarily memory size).

Use the following general guidelines to set the value of the maxusers keyword:

- For systems with limited physical memory and a small number of users, set maxusers to 8 or 16.
- Setting maxusers to 32 is reasonable for most systems with moderate hardware configurations and a moderate number of users.
- For larger systems with heavy workloads, set maxusers to 64.

When you modify maxusers, you also modify the value of a number of other keywords that are based upon maxusers. The keywords that are based on maxusers are all keywords that control system resources that are needed by users and should, therefore, be raised or lowered depending upon the normal user load for your system. The best way to adjust these keywords is to adjust the maxusers keyword. The following keywords are based on the maxusers keyword:

- maxcallouts
- nclist
- nquota
- nvnode
- ndquot
- task_max

5.6.1.7 Maximum Number of clists

The nclist keyword is based on the maxusers keyword and defines the number of clists available on the system. Each clist is a buffer for terminal I/O. The nclist keyword overrides the default value for nclist in the param.c file. The default value should be sufficient for most configurations. Exceptions include third-party asynchronous boards and layered products that perform terminal emulation.

5.6.1.8 Maximum Number of Open Files

The max_vnodes keyword defines the maximum number of VFS files that can be open at a given time system-wide. Each open file is associated with a vnode. If more vnodes are available on the system, more files can be open. However, more system memory is consumed.

On 24 MB systems, this keyword is defined to 1000 by default. For larger systems, the default value is calculated based on system memory size and the percentage of total memory that can be used for vnodes (5 percent by default).

The following example shows the max_vnodes keyword set to 1000:

```
max_vnodes    1000
```

5.6.1.9 Maximum Number of Threads

The maxuthreads keyword defines the maximum number of threads per task. This limit applies to nonprivileged tasks. A task running with superuser privilege can exceed the maxuthreads limit.

5.6.1.10 Maximum Number of System Threads

The threadmax keyword defines the maximum number of threads that can be allocated on the system. This limit is systemwide. The following message is displayed if the system reaches the threadmax limit while creating a new process:

```
fork/procdup: thread_create failed. Code: 6
```

5.6.1.11 Maximum Number of Processes

The task_max keyword is based on the maxusers keyword and sets a limit on the number of processes that can be running on the system. Normally, you should modify the maxusers keyword, rather than the task_max keyword. Initially, task_max is set to the following:

```
1+ (20 + (8 * maxusers))
```

This value is not absolute. It is used to determine the size of a data structure that controls the number of user processes that can run simultaneously. Increasing the value of task_max allows more user processes to be active at the same time. Decreasing this value limits the number of user processes.

The system displays the following message if it reaches the task_max limit:

```
pid: table is full
```

You can find the previous message in the /var/adm/messages file and in the kernel event-logging file.

The task_max kernel parameter in the param.c file is equivalent to the task_max keyword in the configuration file.

5.6.1.12 Maximum Number of User Processes

The maxuprc keyword defines the maximum number of processes one user can run simultaneously. A task running with superuser privilege can exceed the maxuprc limit.

5.6.1.13 Maximum Number of Callouts

The maxcallouts keyword is based on the maxusers keyword and defines the maximum number of callouts on the system. It is used to size the kernel's callout table. The default number of callouts is determined automatically based on the value of the maxusers keyword and other system parameters. Use of the default maxcallouts definition is strongly recommended.

In the unlikely event that the default value of maxcallouts is not large enough, your system will panic with a "timeout table overflow" message. To override the default number of callouts, use the following syntax to add the maxcallouts keyword to your configuration file:

maxcallouts [*number*]

To determine the correct value for *number*, you need to understand the maxcallouts sizing algorithm and to find the current number of callouts. To examine the sizing algorithm, edit the /usr/sys/conf/param.c file. Search for the string MAXCALLOUTS, and print the next several lines. You will notice the algorithm differs for a realtime kernel. To determine the current number of callouts, enter the following commands:

```
# dbx -k /vmunix
(dbx) p ncallout
1316
(dbx) q
```

5.6.1.14 File System Metadata Cache Size

Digital UNIX utilizes a unified buffer cache (UBC). The UBC enables physical memory to be shared between the file system and virtual memory. The Advanced File System (AdvFS) uses UBC. However, the file system code that deals with the UNIX File System (UFS) metadata (including directories, indirect blocks, and inodes) uses the traditional BSD buffer cache.

The bufcache keyword defines the size of the kernel's metadata cache. The value for bufcache is the percentage of the system's physical memory that is allocated for the metadata cache. The default metadata cache memory allocation is 3% of physical memory.

Note that any additional memory that you allocate to the metadata cache is taken away from the rest of the system. This means that the memory is not available to the UBC that caches file data and virtual memory data and is involved in running processes. If you allocate too much memory to the metadata cache, system performance may decline.

5.6.1.15 Machine Architecture

The `machine` keyword defines the architecture of the machine on which the kernel will run. For example:

```
machine    alpha
```

5.6.1.16 Machine Type

The `cpu` keyword defines the specific architectural machine type on which the kernel will run. For example:

```
cpu          "DEC3000_400"
```

5.6.1.17 System SCS Identifier

The `scs_sysid` keyword identifies each device on the CI to the SCS subsystem. The devices supported on the CI are RA class devices.

The argument must be a unique value. During the installation, a value is automatically included in the configuration file. If a CI is not detected during installation, the default value 1 is used.

5.6.1.18 Virtual Memory

You can use the following parameters to tune virtual memory:

`heappercent` — Specifies the percentage of kernel virtual address space to allocate for use by the heap. The default is 7%.

`kentry_zone_size` — Specifies the amount of kernel virtual address space that is available to create kernel virtual address map entries. The default is 16777216, which is adequate for most system environments. If the system panics and issues a message indicating that the `kentry_zone_size` parameter value is too small to support the current workload, increase the parameter value until it is sufficient for the workload.

`mapentries` — Specifies the maximum number of virtual memory map entries. The default is 200.

`maxvas` — Specifies the maximum virtual address space for a user map. The default is 1073741824.

segmentation	Enables or disables shared page tables. The default is 1 (enabled).
swapbuffers	Specifies the maximum number of swap buffers that are available for swap I/O. The default is 128.
ubcbuffers	Specifies the minimum number of buffers that the unified buffer cache can contain. The default is 256.
vpagemax	Specifies the maximum vpage for user map, or the maximum number of individually protected pages. The default is 16384.
zone_size	Specifies the amount of kernel virtual address space that is available for many of the system's dynamic data structures. The default is 67108864, which is sufficient for most system environments. If the system panics and issues a message indicating that the zone_size value is too small to support the current workload, increase the parameter value until it is sufficient for the workload.

5.6.2 System Definition Keyword

The config keyword defines the kernel configuration in terms of the location of the root file system and the dump and swap areas. The recommended config keyword entry selects the a partition of the disk from which the kernel was booted as the root file system. For example:

```
config vmunix swap generic
```

You can use the swapon command to allocate swap areas from the /etc/fstab file. By default, the system assigns the dump area to the same partition as the first swap area found in the /etc/fstab file. You should use the default dump area; however, you can override the default by adding the dumps keyword to the config specification.

For example, to dump to the b partition of an RZ-class disk configured as unit 1, add the following line to the configuration file:

```
config  vmunix  root on rz1a  dumps on rz1b
```

You must also list rz1b as a swap device in the /etc/fstab file, as shown:

```
/dev/rz1b          dump1        ufs     sw      0       2
```

Note that the kernel will unconditionally write the crash dump to `rz1b`, thus destroying any data on that partition. In most cases, crash dumps should be written to one of the swap partitions. For more information about controlling how the system writes crash dumps, see *Kernel Debugging*.

When you specify the `dumps` keyword, you also need to specify the location of the root file system with the `root on` keyword. In the previous example, the root file system is located on the `a` partition of the `rz1` disk. The root file system must be located on the specified partition, otherwise the system will not boot.

5.6.3 Device Definition Keywords

The configuration file contains entries that define hardware devices for your system. These entries include buses, controllers, and storage devices. When your system is initially configured, the `sizer` program identifies all the devices physically attached to your system and places their associated entries in the configuration file.

For a complete list of supported devices, refer to the `GENERIC` configuration file, the *Software Product Description* (SPD), or Appendix A.

5.6.4 The callout Keyword Definitions

The `callout` keyword definitions allow you to run any shell command subprocess during kernel configuration. The `callout` keyword definition invokes a subprocess, and the `config` program waits for the subprocess to complete before continuing the configuration. For example, you can define a `callout` keyword to send mail at a specific time during the configuration.

You can invoke any function with a `callout` keyword definition. If you use a `callout` keyword, you must make sure that the command is in the search path or that the full pathname is specified. Also, any system resources required, such as memory, disks, or tapes, must be available. There is no mechanism for determining if a subprocess succeeds or fails; the `config` command behaves as if the subprocess succeeded. The function must handle its own error conditions.

The `callout` keyword definition specifies the point in the configuration sequence at which to invoke the subprocess. The `CONFIG_NAME` environment variable specifies the configuration file that is used as an argument to the `config` command. The subprocess that is called out uses the environment variable to identify the configuration file.

The following table describes the `callout` keyword definitions and the times at which they are invoked by the `config` command:

Definition	Time Invoked
at_start	After the config command has parsed the configuration file syntax but before processing any input
at_exit	Immediately before the config command exits, regardless of its exit status
at_success	Before the at_exit process, if specified, and only if the config command exits with a success exit status
before_h	Before the config command creates any *.h files
after_h	After the config command creates any *.h files
before_c	Before the config command creates any *.c files
after_c	After the config command creates any *.c files
before_makefile	Before the config command creates the Makefile file
after_makefile	After the config command creates the Makefile file
before_conf	Before the config command creates the conf.c file
after_conf	After the config command creates the conf.c file

More than one callout keyword with the same parameter can be in the configuration file, and the subprocesses are invoked in the order that they appear in the file. The following is an example of some configuration file entries:

```
callout at_exit "echo Exit 1 | mail root"
callout at_exit "echo Exit 2 | mail root"
callout at_success "echo Configuration successful | mail root"
```

If the config command exits successfully, the sequence of the information mailed to root is as follows:

1. Configuration successful
2. Exit 1
3. Exit 2

5.6.5 The options Keyword Definitions

The configuration file contains several definitions that are preceded by the options keyword. In general, these definitions specify the components that define the kernel itself, the subsystems, and additional functionality that is required for the target system to operate correctly. These dependency options usually are mandatory and should not be removed from the configuration file or altered. See Table 5–2 for a complete list of dependency options.

5.6.5.1 Symmetrical Multiprocessing

You do not have to add special configuration options for symmetrical multiprocessing (SMP). The system determines at boot time whether it has multiple CPUs and configures itself accordingly. The default for multiprocessor systems is to configure SMP. For information on how to override this default, see *System Tuning and Performance Management*.

The following mandatory configuration file options have been added to support SMP:

Option Definition	Required
SER_COMPAT	Yes
UNIX_LOCKS	Yes

5.6.5.2 Real-Time Processing

You can enable real-time preemption on your system by defining the RT_PREEMPT_OPT keyword. When this keyword is defined, the system interrupts lower priority processes more often than normal in favor of higher priority processes.

Defining this keyword might degrade the throughput performance of your system because the system spends more time context-switching than it does when you omit the RT_PREEMP_OPT keyword from the configuration file.

Other, required kernel options keywords that are related to real-time processing are as follows:

- RT_PML
- RT_PREEMPT_OPT
- RT_SCHED
- RT_SCHED_RQ
- RT_SEM
- RT_TIMER

Do not remove these required options keywords from your configuration file.

5.6.5.3 Maximum Size of Switch Tables

To accommodate loadable subsystems, you might need to increase the number of entries in the block and character switch tables. The following example shows an error message from the config program that indicates the need for more entries in the block and character switch tables:

```
cfe: Error: conf.c, line 925: Too many initial values for 'bdevsw'
```

If you receive a message similar to this one, edit the the configuration file and define the `options` keywords `MAX_BDEVSW` and `MAX_CDEVSW`. For example, the following line sets the `MAX_BDEVSW` keyword to 64:

```
options  MAX_BDEVSW=64
```

Refer to Table 5–2 for information about the default values for these keywords.

5.6.5.4 File System Configuration

The operating system views file systems as kernel subsystems. The file systems supported by your system are listed in the configuration file using options keywords, as follows:

Options Keyword	Required	Use
BUFCACHE_STATS	Yes	File system statistics gathering
CDFS	No	ISO 9660 CDFS
COMPAT_43	Yes	Backward compatibility with 4.3BSD
QUOTA	Yes	UFS disk quota functionality

5.6.5.5 File System Types, File Formats, and Locking

The following configuration file entries define code dependencies for the supported file system types. Include in your configuration file the entries that apply to your configuration:

Option Definition	Required	Use
CDFS	Yes	ISO 9660 compact disk file system
COMPAT_43	Yes	4.3BSD backwards compatibility
FFM_FS	No	File-on-File File System; needed for STREAMS `fattach` and `mkfifo`
LABELS	Yes	OSF/1 disk labels
MSFS	No	Advanced File System (AdvFS)
NFS	Yes	Network File System (NFS)
NFS_SERVER	No	Server for NFS
OSF_MACH_O	Yes	Format of load files
PROC_FS	No	`/proc` file system (used by `DECladebug`)

Option Definition	Required	Use
QUOTA	Yes	Disk quotas
SYSV_COFF	Yes	Format of load files: System V COFF executables
SYSV_ELF	Yes	System V
SYSV_FS	No	System V File System
UFS	Yes	UNIX File System
UNIX_LOCKS	No[a]	Locking version (parallel)

[a]Yes if a realtime kernel.

5.6.5.6 Standard Digital UNIX Kernel Features and Dependencies

The following configuration file entries define some of the features and dependencies that relate to the Digital UNIX kernel: In your configuration file, include those entries that define the requirements of your configuration:

Options Keyword	Required	Use
OSF	Yes	OSF/1 kernel
GENERIC	Yes	Generic kernel
MACH	Yes	Standard Mach features
MACH_CO_INFO	Yes	Call-out queue information
MACH_COMPAT	Yes	Vendor syscall compatibility
MACH_DEVICE	Yes	Mach I/O support
MACH_IPC_STATS	Yes	Collect IPC statistics
MACH_IPC_TCACHE	Yes	IPC translation cache
MACH_IPC_WWA	Yes	IPC wakeup-when-available
MACH_IPC_XXXHACK	Yes	Mach IPC backward compatibility
MACH_NET	Yes	Fast network access
MACH_SCTIMES	Yes	Dummy system calls for timing
ULT_BIN_COMPAT	Yes	Enables ULTRIX binary compatibility

5.6.5.7 Remote Kernel Debugging

The KDEBUG keyword controls your ability to use the dbx -remote command. If your kernel is built with KDEBUG, you can debug the running kernel using dbx -remote.

5.6.5.8 Network Time Protocol Daemon

The NTP_TIME keyword enables the kernel phase lock loop (PLL) time adjusting algorithm. This algorithm is described by the DARPA Network Working Group Report RFC-1589, for use with the Network Time Protocol (NTP) V3 daemon.

5.6.5.9 Autonice Threads Prioritizing

The AUTONICE keyword lowers the priority of threads that exceed 10 minutes of CPU user time. It does this by automatically "nicing" up the priority of the thread by 4. By default, the AUTONICE feature is off. You should enable this feature if you want threads that run for a long time to have their priority lowered, relative to other threads. You should not enable this feature if you routinely run threads that accumulate significant amounts of CPU time and do not want the priority of these threads lowered.

5.6.5.10 Statistics Functionality

The following configuration file entries define the code dependencies that enable statistics-gathering functionality. In your configuration file, include the entries that you need:

Options Keywords	Required	Use
BUFCACHE_STATS	Yes	Buffer cache statistics
INOCACHE_STATS	Yes	Inode cache statistics
STAT_TIME	Yes	Use statistical timing
VAGUE_STATS	Yes	Vague counts (parallel)

5.6.5.11 Network and Communications Protocols and Dependencies

The following configuration file entries define the code dependencies for network and communications functionality. In your configuration file, include the entries that correspond to the network functionality at your site:

Options Keywords	Required	Use
DLI	No	Data Link Interface
DLPI	No	Data Link Provider Interface
INET	No	Internet protocols
LAT	Yes	LAT protocols
PACKETFILTER	No	Kernel packetfilter support
PPP	No	Point-to-Point Protocol (PPP) for TCP/IP support
SL	No	Serial Line Interface Protocol (SLIP) for TCP/IP support
STREAMS	Yes	STREAMS Framework
STRKINFO	Yes	STREAMS kernel information
TIMOD	No	Transport Interface Streams Module
TIRDWR	No	Transport Interface Read/Write Streams Module
TRN	No	Token Ring Network support for DECnet
TRSRCF	No	Token Ring Source Routing Module
UIPC	Yes	UNIX domain sockets
XTISO	No	X/Open Transport Interface

The DLPI option is dependent on the DLI option. Therefore, if you select the DLPI option, you must also select the DLI option. The DLI option is not dependent on the DLPI option.

Selection of the DLPI option configures the Datalink Bridge Driver (DLB), which implements a partial subset of the DLPI specification. See the *Network Programmer's Guide* for more information.

The PPP option is dependent upon the INET option. The number of PPP lines is configurable using the nppp parameter in the /etc/sysconfigtab file. The default value for nppp is 1.

The SL option is dependent upon the INET option. The number of SLIP lines is configurable using the nslip parameter in the /etc/sysconfigtab file. The default value for nslip is 1.

The TRSRCF option is for Token Ring driver developers who want to use the Token Ring Source Routing functionality for the extended Token Ring Network. See the *Network Programmer's Guide* for more information.

5.6.5.12 Terminal Subsystem

The following configuration file entries define the code dependencies for terminal subsystems. To determine which terminal subsystem is configured into your kernel, include an entry from the following table in your configuration file:

Options Keyword	Required	Use
BSD_TTY	Yes	Berkeley (`clist-based`) TTY
LDTTY	No	STREAMS-based TTY
PCKT	No	STREAMS packet module

5.6.6 The makeoptions Keywords

Certain options are passed to the compiler, assembler, and linker when the kernel is built. These options are identified with the `makeoptions` keywords and take an argument of the form *argument= value*.

```
# makeoptions     ASOPTS="-w"
makeoptions     CDEBUGOPTS="-g3"
makeoptions     PROFOPTS="-DPROFILING -DPROFTYPE=4"
makeoptions     LOADADDR="ffffffff00000000"
# makeoptions     LDOPTS="-x"
```

Note

The `ASOPTS=-w` makeoption is commented out because using it disables C compiler warning messages.

5.6.7 The pseudo-device Keywords

The configuration file contains several keywords that are categorized under the broad `pseudo-device` keyword. These include terminal services, the Logical Storage Manager (LSM), and additional network protocol families and services definitions. The configuration file must contain definitions that correspond to the network protocols and services upon which file systems and communications services depend. The following sections list the related dependencies that are defined with the `pseudo-device` keyword.

See Table 5–2 for a complete list of pseudo-device keyword definitions.

Refer to the *Network Administration* manual for detailed information about supported network and communications services.

5.6.7.1 Mandatory Definitions

The following `pseudo-device` keyword definitions are required in the configuration file:

```
pseudo-device cpus 1
pseudo-device rt_hab
```

The following keyword definition must be included if you want the System V habitat:

```
pseudo-device sysv_hab
```

The following definition must be included if you want the ULTRIX compatibility module:

```
pseudo-device ult_bin
```

5.6.7.2 Graphics

The following `pseudo-device` keyword definitions are required to enable graphic device support for workstations:

```
pseudo-device ws
pseudo-device xcons
```

5.6.7.3 Prestoserve

If your system is equipped with the Prestoserve hardware, the `/usr/sbin/doconfig` program includes the following entry during the initial system configuration:

```
pseudo-device    presto
```

The previous entry must be present in the configuration file for Prestoserve to operate properly.

Certain Prestoserve hardware implementations require an additional entry in the system configuration file. For information on the Prestoserve hardware and its supporting software, see the *Guide to Prestoserve*.

5.6.7.4 Terminal Service

The `pty` configuration file entry specifies the number of available pseudo-ttys (used for incoming network logins and for windows). Define

this entry according to the maximum number of ptys supported by your configuration. You can set the value to any number greater than 16 and less than or equal to 3162. The number of logins (users) is not the same as the number of pseudo-ttys.

There are two implementations of `ptys` available: STREAMS-based and `clist`-based. The default is STREAMS-based and is specified with the `RPTY` keyword. You set the number of psuedo-ttys in the `/etc/sysconfigtab` file as follows:

```
pts:
   nptys = 512
```

The default is 255.

Note that you must also have the `LDTTY` option defined for STREAMS-based ptys.

You define the `clist`-based implementation as follows:

```
pseudo-device   pty   255
```

You may use either the `RPTY` option or the `pty` psuedodevice, but not both.

5.6.7.5 Logical Storage Manager

The Logical Storage Manager (LSM) expands and enhances the standard UNIX system mechanism for data storage, data retrieval, and data protection.

Note

The Logical Volume Manager was retired in Digital UNIX Version 4.0. To continue using enhanced disk management software, you must migrate to the Logical Storage Manager (LSM) software. Information about migration to LSM using a process called encapsulation is provided in the document *Logical Storage Manager*.

LSM provides a virtual disk that enables you to store and replicate data without physical boundaries. LSM is composed of physical devices and logical entities to offer you a mechanism for transparently and dynamically storing and retrieving files and file systems across devices and in multiple copies.

If you perform an Advanced Installation and select the LSM subsets, the following line that enables LSM is automatically placed in your target configuration file:

```
pseudo-device   lsm     1
```

When the /usr/sbin/doconfig runs during installation, LSM is built into the kernel.

However, if you install the LSM subsets after installation with the setld utility, you must add LSM to the kernel by following these steps:

1. Run the /usr/sbin/doconfig program without any options, as described in Section 5.4.2.
2. Select the LSM kernel option from the KERNEL OPTIONS menu to add LSM to the target configuration file and build a new kernel.
3. To enable LSM, you boot the new kernel, as described in Section 5.4.2.

5.6.7.6 Ethernet ARP

If your system uses Ethernet Address Resolution Protocol (ARP) hardware, the /usr/sbin/doconfig program includes the following entry during initial system configuration. This entry must be present in the configuration file:

```
pseudo-device    ether
```

5.6.7.7 Gateway Screen

If you set your system up as a router and you plan to use the gateway screen feature, add the following line to your system configuration file:

```
pseudo-device gwscreen
```

For more information on the gateway screen, see the screend(8) reference page.

5.6.7.8 Packetfilter

To configure the kernel packetfilter device, include the following line in the configuration file:

```
"options PACKETFILTER"
```

5.6.7.9 Network Loopback Device

If your configuration requires the software loopback interface definition, the following entry must be present in the configuration file:

```
pseudo-device     loop
```

5.6.7.10 Additional STREAMS Definitions

If your configuration supports STREAMS protocols, the following entry should be present in the configuration file:

```
pseudo-device     strpush      16
```

The `strpush` entry specifies the number of modules that you can push in a single stream.

6

Administering Devices with Dynamic Device Recognition

This chapter describes the Dynamic Device Recognition (DDR) database, which you use to administer devices in the SCSI/CAM I/O subsystem. It explains how you use the `ddr_config` utility to manage the DDR database on your system. This chapter introduces DDR, then describes how you use the `ddr_config` utility to:

- Add SCSI devices to the DDR database
- Convert a customized `cam_data.c` file

This chapter also discusses adding pseudoterminals and disks and tapes that are not SCSI devices to the operating system.

6.1 Understanding Dynamic Device Recognition

Dynamic Device Recognition is a framework for describing the operating parameters and characteristics of SCSI devices to the SCSI CAM I/O subsystem. You can use DDR to include new and changed SCSI devices into your environment without having to reboot the operating system. You do not disrupt user services and processes, as happens with static methods of device recognition.

Beginning with Digital UNIX Version 4.0, DDR is preferred over the current, static method for recognizing SCSI devices. The current, static method, as described in Chapter 5, is to edit SCSI device customizations into the `/sys/data/cam_data.c` data file, reconfigure the kernel, and shut down and reboot the operating system.

Note

Support for the static method of recognizing SCSI devices will be retired in a future release of Digital UNIX .

Digital UNIX Version 4.0 supports both methods of recognizing SCSI devices. Both methods can be employed on the same system, with the restriction that the devices described by each method are exclusive to that method (nothing is doubly-defined).

The information DDR provides about SCSI devices is needed by SCSI drivers. You can supply this information using DDR when you add new SCSI devices to the system, or you can use the `/sys/data/cam_data.c` data file and static configuration methods. The information provided by DDR and the `cam_data.c` file have the same objectives. When compared to the static method of providing SCSI device information, DDR minimizes the amount of information that is supplied by the device driver or subsystem to the operating system and maximizes the amount of information that is supplied by the device itself or by defaults specified in the DDR databases.

6.1.1 Conforming to Standards

Devices you add to the system should conform to the SCSI-2 standard, as specified in *SCSI-2, Small Computer System Interface-2 (X3.131-1994).* If your devices do not comply with the standard, or if they require exceptions from the standard, you store information about these differences in the DDR database. If the devices comply with the standard, there is usually no need to modify the database.

6.1.2 Understanding DDR Messages

Following are the most common DDR message categories and the action, if any, that you should take.

- Console messages are displayed during the boot sequence.

  Frequently, these messages indicate that the kernel cannot read the DDR database. This error occurs when the system's firmware is not at the proper revision level. Upgrade to the correct revision level of the firmware.

- Console messages warn about corrupted entries in the database. Recompile and regenerate the database.

- Runtime messages generally indicate syntax errors that are produced by the `ddr_config` compiler. The compiler runs when you use the `–c` option to the `ddr_config` utility and does not produce an output database until all syntax errors have been corrected.

6.1.3 Getting Help with ddr_config Options

Use the `–h` option to the `ddr_config` command to display help on command options.

6.2 Changing the DDR Database

When you make a change to the operating parameters or characteristics of a SCSI device, you must describe the changes in the `/etc/ddr.dbase` file. You must compile the changes by using the `ddr_config -c` command.

Two common reasons for changes are:

- Your device deviates from the SCSI standard or reports something different from the SCSI standard
- You want to optimize device defaults, most commonly the `TagQueueDepth` parameter, which specifies the maximum number of active tagged requests the device supports

You use the `ddr_config -c` command to compile the `/etc/ddr.dbase` file and produce a binary database file, `/etc/ddr.db`. When it is notified that the file's state has changed, the kernel loads the new `/etc/ddr.dbase` file. In this way, the SCSI CAM I/O subsystem is dynamically updated with the changes that you made in the `/etc/ddr.dbase` file and the contents of the on-disk database are synchronized with the contents of the in-memory database.

Use the following procedure to compile the `/etc/ddr.dbase` database:

1. Log in as root or become the superuser.
2. Enter the `ddr_config -c` command, for example:

```
# /sbin/ddr_config -c
#
```

When the prompt is displayed, the compilation is complete. If there are syntax errors, they are displayed at standard output and no output file is compiled.

6.3 Converting Customized cam_data.c Information

You use the following procedure to transfer customized information about your SCSI devices from the `/sys/data/cam_data.c` file to the `/etc/ddr.dbase` text database. In this example, *MACHINE* is the name of your machine's system configuration file.

1. Log on as root or become the superuser.
2. To produce a summary of the additions and modifications that you should make to your `/etc/ddr.dbase` file, enter the `ddr_config -x` command. For example:

```
# /sbin/ddr_config -x MACHINE > output.file
```

The command uses as input the system configuration file that you used to build your running kernel. The procedure runs in multiuser mode and requires no input after it has been started. You should redirect output to a file in order to save the summary information. Compile errors are reported to standard error and the command terminates when the error is reported. Warnings are reported to standard error and do not terminate the command.

3. Edit the characteristics that are listed on the output file into the `/etc/ddr.dbase` file, following the syntax requirements of that file. Instructions for editing the `/etc/ddr.dbase` database are found in `ddr.dbase`(4).
4. Enter the `ddr_config -c` command to compile the changes.

See Section 6.2 for more information.

6.4 Adding Pseudoterminals and Devices without Using DDR

You can add pseudodevices, disks, and tapes statically, without using DDR, by using the methods described in the following sections.

6.4.1 Adding Pseudoterminals

Pseudoterminals enable users to use the network to access a system. A pseudoterminal is a pair of character devices that emulates a hardware terminal connection to the system. Instead of hardware, however, there is a master device and a slave device. Pseudoterminals, unlike terminals, have no corresponding physical terminal port on the system. Remote login sessions, window-based software, and shells use pseudoterminals for access to a system. Digital UNIX offers two implementations of pseudoterminals: BSD STREAMS and BSD `clist`.

For some installations, the default number of `pty` devices is adequate. However, as your user community grows, and each user wants to run multiple sessions of one or more timesharing machines in your environment, the machines may run out of available `pty` lines.

To add pseudoterminals to your system:

1. Log in as root.
2. Edit the pseudodevice entry in the system configuration file. By default, the kernel supports 255 pseudoterminals. If you add more pseudoterminals to your system, you must edit the system configuration file entry and increment the number 255 by the number

of pseudoterminals you want to add. The following examples show that 400 pseudoterminals have been added.

The pseudodevice entry for STREAMS-based pseudoterminals is as follows:

```
pseudo-device rpty 655
```

The pseudodevice entry for `clist-` based pseudoterminals is as follows:

```
pseudo-device pty 655
```

For more information on the configuration file and its its pseudodevice keywords, refer to Chapter 5.

3. Rebuild and boot the new kernel. Use the information on rebuilding and booting the new kernel in Section 5.4.3.

   When the system is first installed, the configuration file contains a pseudodevice entry with the default number of 255 pseudoterminals. If for some reason the number is deleted and not replaced with another number, the system defaults to supporting 80 pseudoterminals.

4. Log in as root and change to the `/dev` directory.

5. Create the device special files by using the `MAKEDEV` command, which has the following syntax:

   ./MAKEDEV pty*#*

   The number sign (#) represents the set of pseudoterminals (0 to 101) you want to create. The first 51 sets (0 to 50) create 16 pseudoterminals for each set. The last 51 sets (51 to 101) create 46 pseudoterminals for each set. You can use the following syntax to create a large number of pseudoterminals:

   ./MAKEDEV PTY_*#*

   The number sign (#) represents the set of pseudoterminals (1 to 9) you want to create. Each set creates 368 pseudoterminals, except the `PTY_3` and `PTY_9` sets, which create 356 and 230 pseudoterminals, respectively.

   Refer to the Software Product Description (SPD) for the maximum number of supported pseudoterminals.

Note

By default, the installation software creates device special files for the first two sets of pseudoterminals, `pty0` and `pty1`. The `pty0` pseudoterminals have corresponding device special files named `/dev/ttyp0` through `/dev/ttypf`. The

pty1 pseudoterminals have corresponding device special files named /dev/ttyq0 through /dev/ttyqf.

If you add pseudoterminals to your system, the pty# variable must be higher than pty1 because the installation software sets pty0 and pty1. For example, to create device special files for a third set of pseudoterminals, enter:

```
# ./MAKEDEV pty2
```

The MAKEDEV command lists the device special files it has created. For example:

```
MAKEDEV: special file(s) for pty2:
ptyr0 ttyr0 ptyr1 ttyr1 ptyr2 ttyr2 ptyr3 ttyr3 ptyr4 ttyr4
ptyr5 ttyr5 ptyr6 ttyr6 ptyr7 ttyr7 ptyr8 ttyr8 ptyr9 ttyr9
ptyra ttyra ptyrb ttyrb ptyrc ttyrc ptyrd ttyrd ptyre ttyre
ptyrf ttyrf
```

6. If you want to allow root logins on all pseudoterminals, make sure an entry for ptys is present in the /etc/securettys file. If you do not want to allow root logins on pseudoterminals, delete the entry for ptys from the /etc/securettys file. For example (using the sample output shown in step 5), to add the entries for the new tty lines and to allow root login on all pseudoterminals, enter the following lines in the /etc/securettys file:

```
/dev/tty08      # direct tty
/dev/tty09      # direct tty
/dev/tty10      # direct tty
/dev/tty11      # direct tty
ptys
```

Refer to the securettys(4) reference page for more information.

The pty name space in SVR4 systems is defined as follows:

/dev/pts/*N*

The variable *N* is a number from 0-9999.

This name space allows for more scalability than the BSD pty name space (tty[a-zA-Z][0-9a-zA-Z]). The base system commands and utilities have been modified to support both SVR4 and BSD pty name spaces. For binary compatibility reasons, the default is the BSD name space. You can alter this behavior by using the SYSV_PTY(8) command. The invocation of the SVS_PTY command results in using the SVR4 name space as the default. To revert back to the original default behavior (BSD pty name space), create the BSD ptys as discussed in Section 6.4.1.

6.4.2 Adding Disk and Tape Drives

When you add new tape or disk drives to your system, you must physically connect the devices and then make the devices known to the system. There are two methods, one for static drivers and another for loadable drivers.

Note

You will need the documentation that came with your system's hardware. This includes such documentation as the owner's guide, the disk drive guide, and the options guide.

To add a device for a loadable driver, see *Writing Device Drivers: Tutorial*.

To add a device for a static driver, see Section 5.4.1.

Next, you make the device special files for the device, by following these steps:

1. Change to the `/dev` directory.
2. Create the device special files by using the `MAKEDEV` command. If you are configuring a RAID controller, follow the procedure described in the `SCSI`(8) reference page. Use the following syntax to invoke the `MAKEDEV` command:

   ./MAKEDEV *device#*

   The *device* variable is the device mnemonic for the drive you are adding. Appendix A lists the device mnemonics for all supported disk and tape drives. The number sign (#) is the number of the device, 0 through 127 for SCSI disk and tape devices. For example, to create the device special files for two SCSI disk drives, enter the following command:

   ```
   # ./MAKEDEV rz5 rz7
   MAKEDEV: special file(s) for rz5:
   rz5a rrz5a rz5b rrz5b rz5c rrz5c rz5d rrz5d rz5e rrz5e rz5f
   rrz5f rz5g rrz5g rz5h rrz5h
   MAKEDEV: special file(s) for rz7:
   rz7a rrz7a rz7b rrz7b rz7c rrz7c rz7d rrz7d rz7e rrz7e rz7f
   rrz7f rz7g rrz7g rz7h rrz7h
   ```

3. Stop system activity by using the `shutdown` command and then turn off the processor. Refer to Chapter 3 for more information.
4. Power up the machine. To ensure that all the devices are seen by the system, power up the peripherals before powering up the system box.
5. Boot the system with the new kernel. Refer to Chapter 3 for information on booting your processor.

7

Administering the UNIX File System

This chapter introduces file systems, disk partitions, and swap space, and explains how to perform the following system administration tasks related to the UNIX File System (UFS):

- Adding swap space
- Managing file system directories and files
- Configuring file system types and locks
- Creating file systems
- Mounting and unmounting file systems
- Tuning and checking file systems
- Managing disk space and troubleshooting disks
- Repartitioning disks
- Cloning disks
- Checking for overlapping partitions on disks

7.1 File Systems and Logical Storage

The Digital UNIX operating system supports many file systems and logical storage schemes, including:

- UNIX File System (UFS)
- POLYCENTER Advanced File System (AdvFS)
- ISO 9660 Compact Disk File System (CDFS)
- Network File System (NFS)
- Virtual File System (VFS) interface and framework
- Logical Storage Manager (LSM)

For a survey of these capabilities, read the Digital UNIX *Technical Overview* manual. For information about administering these file systems, see Chapter 8 for AdvFS and Chapter 9 for LSM.

Administration of the NFS is documented in the *Network Administration* manual.

The *Technical Overview* points you to sources of information about these file systems:

- Memory File System (MFS)
- `/proc` File System (PROCFS)
- File-on-File Mounting File System (FFM)
- File Descriptor File System (FDFS)

7.1.1 Disk Partitions

A disk consists of storage units called sectors. Each sector is usually 512 bytes. A sector is addressed by the logical block number (LBN). The LBN is the basic unit of the disk's user-accessible data area that you can address. The first LBN is numbered 0, and the highest LBN is numbered one less than the number of LBNs in the user-accessible area of the disk.

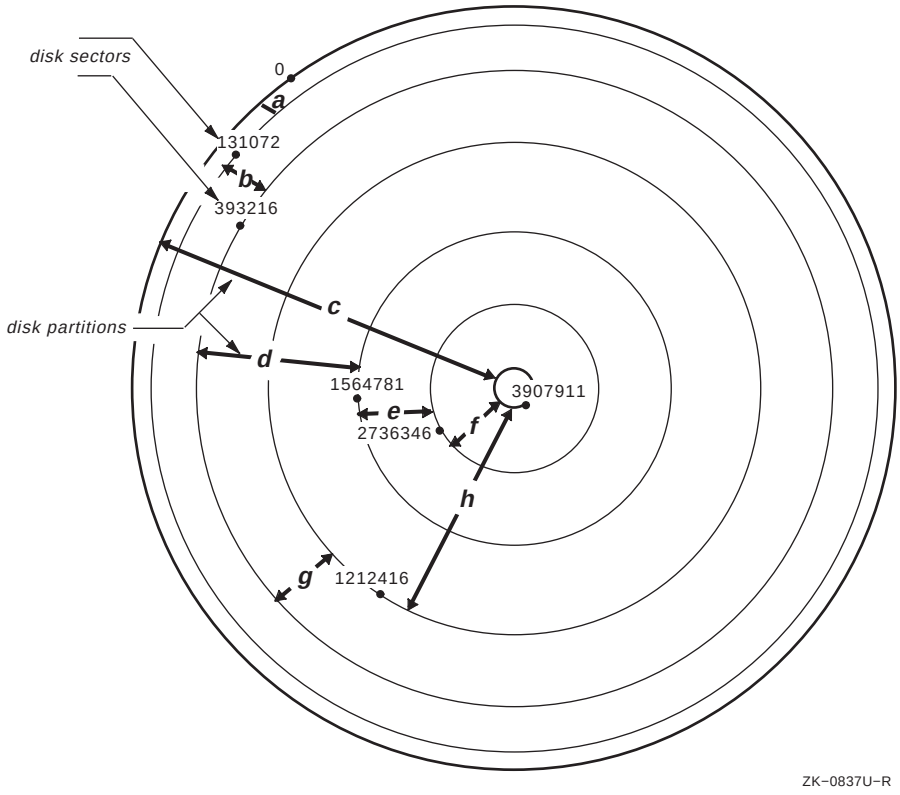
Sectors are grouped together to form up to eight disk partitions. However, disks differ in the number and size of partitions. The `/etc/disktab` file contains a list of supported disks and the default partition sizes for the system. Refer to the `disktab`(4) reference page for more information.

Disk partitions are logical divisions of a disk that allow you to organize files by putting them into separate areas of varying sizes. Partitions hold data in structures called file systems and can also be used for system operations such as paging and swapping. File systems have a hierarchical structure of directories and files, which is described in Section 7.1.3. By selecting the file systems to be placed in a partition, you can monitor the growth and activity of the disk.

Disk partitions have default sizes that depend on the type of disk and that can be altered by using the `disklabel` command. Partitions are named `a` to `h`. While the allocated space for a partition can overlap another partition, a properly partitioned disk should not have file systems on overlapping partitions.

Figure 7–1 shows the default partitions and starting (offset) sectors for an RZ73 disk:

Figure 7–1: RZ73 Default Disk Partitions



The disk label is located in block 0 (zero) in one of the first sectors of the disk. The disk label provides detailed information about the geometry of the disk and the partitions into which the disk is divided. The system disk driver and the boot program use the disk label information to recognize the drive, the disk partitions, and the file systems. Other information is used by the operating system to use the disk most efficiently and to locate important file system information.

The disk label description of each partition contains an identifier for the partition type (for example, standard file system, swap space, and so on). There are two copies of a disk label, one located on the disk and one located in system memory. Because it is faster to access system memory than to perform I/O, when a system recognizes a disk, it copies the disk label into memory. The file system updates the in-memory copy of the label if it contains incomplete information about the file system. If a disk is not labeled and does not contain an ULTRIX-style partition table, the partitions are obtained from the `/etc/disktab` file. You can change the

label with the `disklabel` command. Refer to Section 7.8 and to the `disklabel`(8) reference page for more information.

7.1.2 Adding Swap Space

The Digital UNIX operating system uses a combination of physical memory and disk space to create virtual memory, which can be much larger than the physical memory. Virtual memory can support more processes than the physical memory alone. This section and the sections that follow describe important virtual memory concepts that you should consider when configuring swap space.

The basic unit of virtual memory and physical memory is the page. Virtual memory attempts to keep a process' most recently referenced virtual pages in physical memory. When a process references virtual pages, they are brought into physical memory from their storage locations on disk. Modified virtual pages can be moved to a temporary location on the disk called swap space if the physical pages (the pages in physical memory) that contain the virtual pages are needed by either a newly referenced virtual page or by a page with a higher priority. Therefore, a process' virtual address space can consist of pages that are located in physical memory, stored temporarily in swap space, and stored permanently on disk in executable or data files. The operating system uses two operations to move virtual pages between physical memory and disk: paging and swapping.

Paging involves moving a single virtual page or a small cluster of pages between disk and physical memory. If a process references a virtual page that is not in physical memory, the operating system reads a copy of the virtual page from its permanent location on disk or from swap space into physical memory. This operation is called a pagein. Pageins typically occur when a process executes a new image and references locations in the executable image that have not been referenced before.

If a physical page is needed to hold a newly referenced virtual page or a page with a higher priority, the operating system writes a modified virtual page (or a small cluster of pages) that has not been recently referenced to the swap space. This operation is called modified page writing or a pageout. Note that only modified virtual pages are written to swap space because there is always a copy of the unmodified pages in their permanent locations on disk.

Swapping involves moving a large number of virtual pages between physical memory and disk. The operating system requires a certain amount of physical memory for efficient operation. If the number of free physical pages drops below the system-defined limit, and if the system is unable to reclaim enough physical memory by paging out individual virtual pages or

clusters of pages, the operating system selects a low priority process and reclaims all the physical pages that it is using. It does this by writing all of its modified virtual pages to swap space. This operation is called a swapout. Swapouts typically occur on systems that are memory constrained.

7.1.2.1 How Swap Space is Allocated

Swap space is initially allocated during system installation. You can add swap space after the installation by including swap space entries in the `/etc/fstab` file and then rebooting. Additionally, you can use the `swapon` command to add more swap space–overriding the `/etc/fstab` definitions–until the next time the system is rebooted. Refer to Section 7.4 and the `swapon`(8) reference page for more information.

See Section 7.10 for information about how this command interacts with overlapping partitions. The amount of swap space that your system requires depends on the swap space allocation strategy that you use and your system workload. Strategies are described in the following section.

7.1.2.2 Estimating Swap Space Requirements

There are two strategies for swap space allocation: immediate mode and deferred or over-commitment mode. The two strategies differ in the point in time at which swap space is allocated. In immediate mode, swap space is allocated when modifiable virtual address space is created. In deferred mode, swap space is not allocated until the system needs to write a modified virtual page to swap space.

Note

The operating system will terminate a process if it attempts to write a modified virtual page to swap space that is depleted.

Immediate mode is more conservative than deferred mode because each modifiable virtual page is assigned a page of swap space when it is created. If you use the immediate mode of swap space allocation, you must allocate a swap space that is at least as large as the total amount of modifiable virtual address space that will be created on your system. Immediate mode requires significantly more swap space than deferred mode because it guarantees that there will be enough swap space if every modifiable virtual page is modified.

If you use the deferred mode of swap space allocation, you must estimate the total amount of virtual address space that will be both created and modified, and compare that total amount with the size of your system's

physical memory. If this total amount is greater than the size of physical memory, the swap space must be large enough to hold the modified virtual pages that do not fit into your physical memory. If your system's workload is complex and you are unable to estimate the appropriate amount of swap space by using this method, you should first use the default amount of swap space and adjust the swap space as needed.

You should always monitor your system's use of swap space. If the system issues messages that indicate that swap space is almost depleted, you can use the `swapon` command to allocate additional swap space. If you use the immediate mode, swap space depletion prevents you from creating additional modifiable virtual address space. If you use the deferred mode, swap space depletion may result in one or more processes being involuntarily terminated.

7.1.2.3 Selecting the Swap Space Allocation Method

To determine which swap space allocation method is being used, check for the existence of a soft link named `/sbin/swapdefault`, which points to the primary swap partition. If the `/sbin/swapdefault` file exists, the system uses the immediate method of swap space allocation. To enable the deferred method, rename or delete this soft link.

You may receive the following informational messages when you remove the `/sbin/swapdefault` file and when you boot a system that is using the deferred method:

```
vm_swap_init:  warning sbin/swapdefault swap device not found
vm_swap_init:  in swap over-commitment mode
```

If the `/sbin/swapdefault` file does not exist and you want to enable the immediate method of swap allocation, become the root user and create the file by using the following command syntax:

ln **–s** ../dev/rz*xy* /sbin/swapdefault

The *x* variable specifies the device number for the device that holds the primary swap partition, and the *y* variable specifies the swap partition. Usually, the swap device number is the same as the boot device number, and the primary swap partition is partition `b`.

You must reboot the system for the new method to take effect.

7.1.3 UNIX File System Structure

This section discusses the UNIX File System (UFS). For information on the Advanced File System (AdvFS) structure, refer to Chapter 8.

A UFS file system has four major parts:

- Boot block

  The first block of every file system (block 0) is reserved for a boot, or initialization, program.

- Superblock

  Block 1 of every file system is called the superblock and contains the following information:

  – Total size of the file system (in blocks)

  – Number of blocks reserved for inodes

  – Name of the file system

  – Device identification

  – Date of the last superblock update

  – Head of the free-block list, which contains all of the free blocks (the blocks available for allocation) in the file system

    When new blocks are allocated to a file, they are obtained from the free-block list. When a file is deleted, its blocks are returned to the free-block list.

  – List of free inodes, which is the partial listing of inodes available to be allocated to newly created files

- Inode blocks

  A group of blocks follows the superblock. Each of these blocks contains a number of inodes. Each inode has an associated inumber. An inode describes an individual file in the file system. There is one inode for each possible file in the file system. File systems have a maximum number of inodes; therefore there is a maximum number of files that a file system can contain. The maximum number of inodes depends on the size of the file system.

  The first inode (inode 1) on each file system is unnamed and unused. The second inode (inode 2) must correspond to the root directory for the file system. All other files in the file system are under the file system's root directory. After inode 2, you can assign any inode to any file. You can also assign any data block to any file. The inodes and blocks are not allocated in any particular order.

  If an inode is assigned to a file, the inode can contain the following information:

  – File type

    The possible types are regular, device, named pipes, socket, and symbolic link files.

- File owner

  The inode contains the user and group identification numbers that are associated with the owner of the file.

- Protection information

  Protection information specifies read, write, and execute access for the file owner, members of the group associated with the file, and others. The protection information also includes other mode information specified by the `chmod` command.

- Link count

  A directory entry (link) consists of a name and the inumber (inode number) that represents the file. The link count indicates the number of directory entries that refer to the file. A file is deleted if the link count is zero; the file's inode is returned to the list of free inodes, and its associated data blocks are returned to the free-block list.

- Size of the file in bytes
- Last file access date
- Last file modification date
- Last inode modification date
- Pointers to data blocks

  These pointers indicate the actual location of the data blocks on the physical disk.

• Data blocks

  Data blocks contain user data or system files.

7.1.4 File System and Directory Hierarchy

The standard Digital UNIX system directory hierarchy is set up for efficient organization. It separates files by function and intended use. Effective use of the file system includes placing command files in directories that are in the normal search path as specified by the users' `.profile` or `.login` file, as appropriate. Figure 7–2 shows the major directories in the file system. Not all of the directories in the Digital UNIX hierarchy are shown; you should use those shown in Figure 7–2 to ensure that your product will be portable to other systems. Some of the directories are actually symbolic links.

Figure 7–2: Partial Digital UNIX Directory Hierarchy

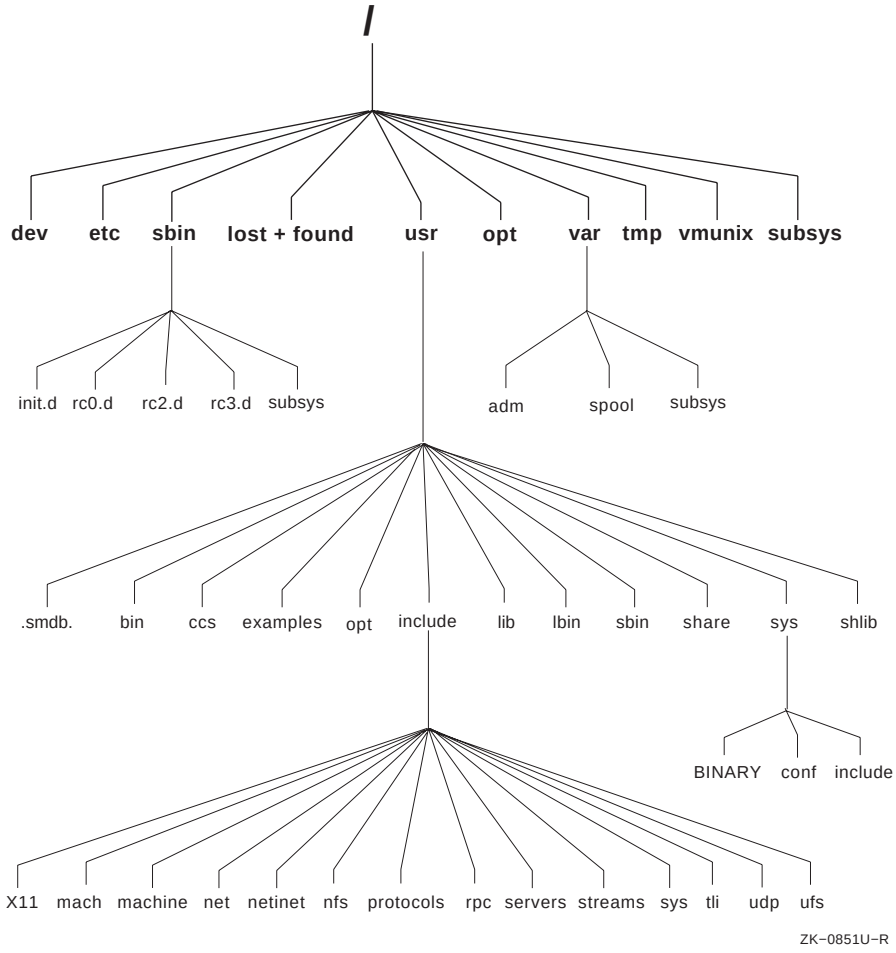


Table 7–1 describes the contents and purposes of the directories shown in Figure 7–2.

Table 7–1: Contents of the Digital UNIX Directories

Directory	Description
`/`	The root directory of the file system.
`dev`	Block and character device files.
`etc`	System configuration files and databases; nonexecutable files.

Table 7–1: Contents of the Digital UNIX Directories (cont.)

Directory	Description
sbin/	Commands essential to boot the system. These commands do not depend on shared libraries or the loader and can have other versions in /usr/bin or /usr/sbin.
init.d	System initialization files.
rc0.d	The rc files executed for system-state 0 (single-user state).
rc2.d	The rc files executed for system-state 2 (nonnetworked multiuser state).
rc3.d	The rc files executed for system-state 3 (networked multiuser state).
subsys	Loadable kernel modules required in single-user mode.
lost+found	Files recovered by fsck.
usr/	Most user utilities and applications. Most of the commands in /usr/bin, /usr/sbin, and /usr/lbin have been built with the shared version of libc and will not work unless /usr is mounted.
.smdb.	Installation control files used by setld.
bin	Common utilities and applications.
ccs	C compilation system; tools and libraries used to generate C programs.
examples	Source code for example programs.
opt	Optional application packages such as layered products.
include/	Program header (include) files; not all subdirectories are listed here.
X11	X11 include files.
mach	Mach-specific C include files.
machine	Machine-specific C include files.
net	Miscellaneous network C include files.
netinet	C include files for Internet standard protocols.
nfs	C include files for NFS.
protocols	C include files for Berkeley service protocols.

Table 7–1: Contents of the Digital UNIX Directories (cont.)

Directory		Description
	rpc	C include files for remote procedure calls.
	servers	C include files for servers.
	streams	C include files for Streams.
	sys	System C include files (kernel data structures).
	tli	C include files for Transport Layer Interface.
	udp	C include files for User Datagram Protocol.
	ufs	C include files for UFS.
lib		Libraries, data files, and symbolic links to library files located elsewhere; included for compatibility.
lbin		Back-end executables.
sbin		System administration utilities and system utilities.
share		Architecture-independent ASCII text files. These files include word lists, various libraries, and online reference pages.
sys		Directories that contain system configuration files.
shlib		Binary loadable shared libraries; shared versions of libraries in /usr/ccs/lib.
opt		Optional application packages such as layered products.
var/		Multipurpose log, temporary, transient, varying, and spool files.
adm		Common administrative files and databases. These files include the crash area, files for the cron daemon, configuration and database files for sendmail, and files generated by syslog.
spool		Miscellaneous printer and mail system spooling directories.
tmp		System-generated temporary files that are usually not preserved across a system reboot.
vmunix		Pure kernel executable (the operating system loaded into memory at boot time).

Mounting a file system makes it available for use. Use the mount command to attach file systems to the file system hierarchy under the system root

directory; use the `umount` command to detach them. When you mount a file system, you specify a location (the mount point under the system root directory) to which the file system will attach.

The root directory of a mounted file system is also its mount point. Only one system root directory can exist because the system uses the root directory as its source for system initialization files. Consequently, file systems are mounted under the system root directory.

7.1.5 Directories and File Types

The operating system views files as bit streams, allowing you to define and handle on-disk data, named pipes, UNIX domain sockets, and terminals as files. This object-type transparency provides a simple mechanism for defining and working with a wide variety of storage and communication facilities. The operating system handles the various levels of abstraction as it organizes and manages its internal activities.

While you notice only the external interface, you should understand the various file types recognized by the system. The system supports the following file types:

- Regular files contain data in the form of a program, a text file, or source code, for example.
- Directories are a type of regular file and contain the names of files or other directories.
- Character and block device files identify physical and pseudodevices on the system.
- UNIX domain socket files provide a connection between network processes. The `socket` system call creates socket files.
- Named pipes are device files that communicating processes operating on a host machine use.
- Linked files point to target files or directories. A linked file contains the name of the target file. A symbolically linked file and its target file can be located on the same file system or on different file systems. A file with a hard link and its target file must be located on the same file system.

7.1.6 Device Special Files

Device special files represent physical devices, pseudodevices, and named pipes. The `/dev` directory contains device special files. Device special files serve as the link between the system and the device drivers. Each device special file corresponds to a physical device (for example, a disk, tape,

printer, or terminal) or a pseudodevice (for example, a network interface, a named pipe, or a UNIX domain socket). The driver handles all read and write operations and follows the required protocols for the device.

There are three types of device files:

- Block device files

  Block device files are used for devices whose driver handles I/O in large blocks and where the kernel handles I/O buffering. Physical devices such as disks are defined as block device files. An example of the block device files in the /dev directory follows:

```
brw-------  1 root  system     8,   1 Jan 19 11:20 /dev/rz0a
brw-------  1 root  system     8,   1 Jan 19 10:09 /dev/rz0b
```

- Character device files

  Character device files are used for devices whose drivers handle their own I/O buffering. Disk, terminal, pseudoterminal, and tape drivers are typically defined as character device files. An example of the character device files in the /dev directory follows:

```
crw-rw-rw-  1 root  system    7,    0 Jan 31 16:02 /dev/ptyp0
crw-rw-rw-  1 root  system    7,    1 Jan 31 16:00 /dev/ptyp1
crw-rw-rw-  1 root  system    9, 1026 Jan 11 14:20 /dev/rmt1h
```

- Socket device files

  The printer daemon (lpd) and error logging daemon (syslogd) use the socket device files. An example of the socket device files in the /dev directory follows:

```
srw-rw-rw-  1 root  system         0 Jan 22 03:40 log
srwxrwxrwx  1 root  system         0 Jan 22 03:41 printer
```

Because disk and tape drivers often handle more than one device, each device file has a major and a minor number. The major number specifies (to the kernel) the driver that handles the device. The minor number is passed to the appropriate driver and tells it the device on which to perform the operation.

For static drivers, use the MAKEDEV command or the mknod command to create device special files. The kmknod command creates device special files for third-party kernel layered products. Refer to the MAKEDEV(8), mknod(8), and kmknod(8) reference pages for more information.

For loadable drivers, the sysconfig command creates the device special files by using the information specified in the driver's stanza entry in the /etc/sysconfigtab database file.

7.2 Creating File Systems

The newfs command formats a disk partition and creates a usable UNIX file system. For information on creating an AdvFS, refer to Chapter 8. Using the information in the disk label or the default values specified in the /etc/disktab file, the newfs command builds a file system on the specified disk partition. You can also use newfs command options to specify the disk geometry.

Note

Changing the default disk geometry values may make it impossible for the fsck program to find the alternate superblocks if the standard superblock is lost.

The newfs command has the following syntax:

/sbin/newfs [–N] [*fs_options*] *device* [*disk_type*]

You must specify the unmounted, raw device (for example, /dev/rrz0a).

Refer to the newfs(8) reference page for information on the command options specific to file systems.

See Section 7.10 for information about how this command interacts with overlapping partitions.

7.3 Checking File Systems

The fsck program checks UNIX file systems and performs some corrections to help ensure a reliable environment for file storage on disks. The fsck program can correct file system inconsistencies such as unreferenced inodes, missing blocks in the free list, or incorrect counts in the superblock.

File systems can become corrupted in many ways, such as improper shutdown procedures, hardware failures, and power outages and power surges. A file system can also become corrupted if you physically write protect a mounted file system, take a mounted file system off line, or if you do not synchronize the system before you shut the system down.

At boot time, the system runs fsck noninteractively, making any corrections that can be done safely. If it encounters an unexpected inconsistency, the fsck program exits, leaves the system in single-user mode, and displays a recommendation that you run the program manually, which allows you to respond yes or no to the prompts that fsck displays.

The command to invoke the fsck program has the following syntax:

/usr/sbin/fsck [*options* ...] [*file_system* ...]

If you do not specify a file system, all the file systems in the `/etc/fstab` file are checked. If you specify a file system, you should always use the raw device.

Refer to the `fsck`(8) reference page for information about command options.

See Section 7.10 for information about how this command interacts with overlapping partitions.

Note

To check the root file system, you must be in single-user mode, and the file system must be mounted read only. To shut down the system to single-user mode, use the `shutdown` command.

AdvFS uses write-ahead logging instead of the `fsck` utility. As your system mounts, AdvFS checks all records in the recovery log for system inconsistencies and makes corrections as needed. Refer to Chapter 8 for more information.

7.4 Accessing File Systems

You attach a file system to the file system tree by using the `mount` command, which makes the file system available for use. The `mount` command attaches the file system to an existing directory (mount point).

Note

The Digital UNIX operating system does not support 4-KB block-size file systems. The default block size for Digital UNIX file systems is 8 KB. To access the data on a disk that has 4-KB block-size file systems, you must back up the disk to either a tape or a disk that has 8-KB block-size file systems.

When you boot the system, file systems that are defined in the `/etc/fstab` file are mounted. The `/etc/fstab` file contains entries that specify the device and partition where the file system is located, the mount point, and additional information about the file system, such as file system type. If you are in single-user mode, the root file system is mounted read only.

Note

To change a file system's mount status, use the `mount` command with the –`u` option. This is useful if you try to reboot and the `/etc/fstab` file is unavailable.

If you try to reboot and the `/etc/fstab` file is corrupted, use a command similar to the following:

```
# mount -u /dev/rz0a /
```

The `/dev/rz0a` device is the root file system.

The operating system uses the UFS for the root file system. The operating system supports only one root file system from which it accesses the executable kernel (`/vmunix`) and other binaries and files that it needs to boot and initialize. The root file system is mounted at boot time and cannot be unmounted.

The `/etc/fstab` file contains descriptive information about file systems and swap space and is read by commands such as the `mount` command. When you boot the system, the `/etc/fstab` file is read and the file systems described in the file are mounted in the order that they appear in the file. A file system or swap space is described on a single line; information on each line is separated by tabs or spaces. Refer to the `swapon`(8) reference page for more information about adding swap space.

The order of entries in the `/etc/fstab` file is important because the `mount` and `umount` commands read the file entries in the order that they appear.

You must be root user to edit the `/etc/fstab` file. To apply the additions that you make to the file, use the `mount –a` command. Any changes you make to the file become effective when you reboot.

The following is an example of an `/etc/fstab` file:

```
/dev/rz2a        /                 ufs     rw      1       1
/dev/rz0g        /usr              ufs     rw      1       2
/dev/rz2b        swap1             ufs     sw      0       2
/dev/rz0b        swap2             ufs     sw      0       2
/dev/rz2g        /var              ufs     rw      1       2
/usr/man@tuscon  /usr/man          nfs     rw,bg   0       0
proj_dmn#testing /projects/testing advfs   rw      0       0
  1                 2               3       4       5       6
```

Each line contains an entry and the information is separated either by tabs or spaces. An `/etc/fstab` file entry has the following information:

1. Specifies the block special device or remote file system to be mounted. For UFS, the special file name is the block special file name, not the character special file name.
2. Specifies the mount point for the file system or remote directory (for example, `/usr/man`) or `swap`*n* for a swap partition.
3. Specifies the type of file system, as follows:

`cdfs`	Specifies an ISO 9600 or HS formatted (CD–ROM) file system.
`nfs`	Specifies NFS.
`procfs`	Specifies a `/proc` file system, which is used for debugging.
`ufs`	Specifies a UFS file system or a swap partition.
`advfs`	Specifies an AdvFS file system.

4. Describes the mount options associated with the partition. You can specify a list of options separated by commas. Usually, you specify the mount type and any additional options appropriate to the file system type, as follows:

`ro`	Specifies that the file system is mounted with read-only access.
`rw`	Specifies that the file system is mounted with read-write access.
`sw`	Specifies that the partition is used as swap space.
`rq`	Specifies that the file system is mounted with read-write access and quotas imposed.

`userquota` `groupquota`	Specifies that the file system is automatically processed by the `quotacheck` command and that disk quotas are enabled with the `quotaon` command. By default, user and group quotas for a file system are contained in the `quota.user` and `quota.group` files, which are located in the directory specified by the mount point. For example, the quotas for the file system on which `/usr` is mounted are located in the `/usr` directory. You also can specify another file name and location. For example: `userquota=/var/quotas/tmp.user`
`xx`	Specifies that the file system entry should be ignored.

5 Used by the `dump` command to determine which UFS file systems should be backed up. If you specify the value 1, the file system is backed up. If you do not specify a value or if you specify 0 (zero), the file system is not backed up.

6 Used by the `fsck` command to determine the order in which the UNIX file system is checked at boot time. For the root file system, specify 1; for other file systems that you want to check, specify 2. If you do not specify a value or if you specify 0 (zero), the file system is not checked. File systems that use the same disk drive are checked sequentially. File systems on different drives are checked simultaneously to utilize the available parallelism.

7.4.1 Using the mount Command

You use the `mount` command to make a file system available for use. Unless you add the file system to the `/etc/fstab` file, the mount will be temporary and will not exist after you reboot the system.

The `mount` command supports the UFS, AdvFS, NFS, CDFS, and `/proc` file system types.

The following `mount` command syntax is for all file systems:

mount [-adflruv] [-o *option*] [-t *type*] [*file_system*] [*mount_point*]

For AdvFS, the file system argument has the following form:

filedomain#fileset

Specify the file system and the mount point, which is the directory on which you want to mount the file system. The directory must already exist on your system. If you are mounting a remote file system, use one of the following syntaxes to specify the file system:

host : *remote_directory*

remote_directory @ *host*

The following command lists the currently mounted file systems and the file system options. The backslash contained in this example indicates line continuation and is not in the actual display.

```
# mount -l
/dev/rz2a on / type ufs (rw,exec,suid,dev,nosync,noquota)
/dev/rz0g on /usr type ufs (rw,exec,suid,dev,nosync,noquota)
/dev/rz2g on /var type ufs (rw,exec,suid,dev,nosync,noquota)
/dev/rz3c on /usr/users type ufs (rw,exec,suid,dev,nosync,noquota)
/usr/share/man@tuscon on /usr/share/man type nfs (rw,exec,suid,dev,
nosync,noquota,hard,intr,ac,cto,noconn,wsize=8192,rsize=8192,
timeo=10,retrans=10,acregmin=3,acregmax=60,acdirmin=30,acdirmax=60)
proj_dmn#testing on /alpha_src type advfs (rw,exec,suid,dev,nosync,\
noquota)
```

The following command mounts the `/usr/homer` file system located on host `acton` on the local `/homer` mount point with read-write access:

```
# mount -t nfs -o rw acton:/usr/homer /homer
```

Refer to the `mount`(8) reference page for more information on general options and options specific to a file system type.

See Section 7.10 for information about how this command interacts with overlapping partitions.

7.4.2 Using the umount Command

Use the `umount` command to unmount a file system. You must unmount a file system if you want to check it with the `fsck` command or if you want to change its partitions with the `disklabel` command. The `umount` command has the following syntax:

umount [-afv] [-h *host*] [-t *type*] [*mount_point*]

If any user process (including a `cd` command) is in effect within the file system, you cannot unmount the file system. If the file system is in use when the command is invoked, the system returns the following error message and does not unmount the file system:

```
mount device busy
```

You cannot unmount the root file system with the `umount` command.

7.5 Tuning File Systems

To enhance the efficiency of UFS reads, use the `tunefs` command to change a file system's dynamic parameters, which affect layout policies.

The tunefs command has the following syntax:

tunefs [-a *maxc*] [-d *rotd*] [-e *maxb*] [-m *minf*] [-o *opt*] [*file_s*]

You can use the tunefs command on both mounted and unmounted file systems; however, changes are applied only if you use the command on unmounted file systems. If you specify the root file system, you must also reboot to apply the changes.

You can use command options to specify the dynamic parameters that affect the disk partition layout policies. Refer to the tunefs(8) reference page for more information on the command options.

7.6 Maintaining Disks

The radisk program and the scu program allow you to maintain your Digital Storage Architecture (DSA) and Small Computer System Interface (SCSI) disk devices, respectively.

With the radisk program, you can perform the following tasks on a DSA disk device:

- Clear a forced error indicator
- Set or clear the exclusive access attribute
- Replace a bad block
- Scan the disk for bad blocks

Refer to the radisk(8) reference page for more information.

The scu program allows you to perform the following tasks on SCSI disk devices, in addition to other tasks:

- Format media
- Reassign a defective block
- Reserve and release a device
- Display and set device and program parameters
- Enable and disable a device

Refer to Appendix B and to the scu(8) reference page for more information.

7.7 Monitoring Disk Use

To ensure an adequate amount of free disk space, you should regularly monitor the disk use of your configured file systems. You can do this in any of the following ways:

- Check available free space by using the `df` command
- Check disk use by using the `du` command or the `quot` command
- Verify disk quotas (if imposed) by using the `quota` command

You can use the `quota` command only if you are the root user.

7.7.1 Checking Available Free Space

To ensure sufficient space for your configured file systems, you should regularly use the `df` command to check the amount of free disk space in all of the mounted file systems. The `df` command displays statistics about the amount of free disk space on a specified file system or on a file system that contains a specified file.

The `df` command has the following syntax:

df [-eiknPt] [-F *fstype*] [*file*] [*file_system*...]

With no arguments or options, the `df` command displays the amount of free disk space on all of the mounted file systems. For each file system, the `df` command reports the file system's configured size in 512-byte blocks, unless you specify the –`k` option, which reports the size in kilobyte blocks. The command displays the total amount of space, the amount presently used, the amount presently available (free), the percentage used, and the directory on which the file system is mounted.

For AdvFS file domains, the `df` command displays disk space usage information for each fileset.

If you specify a device that has no file systems mounted on it, `df` displays the information for the root file system.

You can specify a file pathname to display the amount of available disk space on the file system that contains the file.

Refer to the `df`(1) reference page for more information.

Note

You cannot use the `df` command with the block or character special device name to find free space on an unmounted file system. Instead, use the `dumpfs` command.

The following example displays disk space information about all the mounted file systems:

```
# /sbin/df
Filesystem         512-blks   used  avail capacity Mounted on
/dev/rz2a            30686  21438   6178    77%   /
/dev/rz0g           549328 378778 115616    76%   /usr
/dev/rz2g           101372   5376  85858     5%   /var
/dev/rz3c           394796     12 355304     0%   /usr/users
/usr/share/man@tsts 557614 449234  52620    89%   /usr/share/man
domain#usr          838432 680320  158112   81%   /usr
```

Note

The `newfs` command reserves a percentage of the file system disk space for allocation and block layout. This can cause the `df` command to report that a file system is using more than 100 percent of its capacity. You can change this percentage by using the `tunefs` command with the `–minfree` flag.

7.7.2 Checking Disk Use

If you determine that a file system has insufficient space available, check how its space is being used. You can do this with the `du` command or the `quot` command.

The `du` command pinpoints disk space allocation by directory. With this information you can decide who is using the most space and who should free up disk space.

The `du` command has the following syntax:

/usr/bin/du [-aklrsx] [*directory...* | *filename...*]

The `du` command displays the number of blocks contained in all directories (listed recursively) within each specified directory, file name, or (if none are specified) the current working directory. The block count includes the indirect blocks of each file in 1-kilobyte units, independent of the cluster size used by the system.

If you do not specify any options, an entry is generated only for each directory. Refer to the `du`(1) reference page for more information on command options.

The following example displays a summary of blocks that all main subdirectories in the `/usr/users` directory use:

```
# /usr/bin/du –s /usr/users/*
440     /usr/users/barnam
43      /usr/users/broland
747     /usr/users/frome
6804    /usr/users/morse
```

```
11183   /usr/users/rubin
2274    /usr/users/somer
```

From this information, you can determine that user rubin is using the most disk space.

The following example displays the space that each file and subdirectory in the /usr/users/rubin/online directory uses:

```
# /usr/bin/du -a /usr/users/rubin/online
1 /usr/users/rubin/online/inof/license
2 /usr/users/rubin/online/inof
7 /usr/users/rubin/online/TOC_ft1
16 /usr/users/rubin/online/build
  .
  .
  .
251 /usr/users/rubin/online
```

Note

As an alternative to the du command, you can use the ls –s command to obtain the size and usage of files. Do not use the ls –l command to obtain usage information; ls –l displays only file sizes.

You can use the quot command to list the number of blocks in the named file system currently owned by each user. You must be root user to use the quot command.

The quot command has the following syntax:

/usr/sbin/quot [-c] [-f] [-n] [*file_system*]

The following example displays the number of blocks used by each user and the number of files owned by each user in the /dev/rz0h file system:

```
# /usr/sbin/quot -f /dev/rrz0h
```

Note

The character device special file must be used to return the information, because when the device is mounted the block special device file is busy.

Refer to the quot(8) reference page for more information.

7.7.3 Setting User and Group Quotas for UFS

This section provides information on setting user and group quotas for UFS. As a system administrator, you establish usage limits for user accounts and for groups by setting file system quotas, also known as disk quotas, for them. For information on setting AdvFS user and group quotas, refer to Chapter 8. For more information on user and group quotas on UFS, refer to Section 10.3.4.

You can apply quotas to file systems to establish a limit on the number of blocks and inodes (or files) that a user account or a group of users can allocate. You can set a separate quota for each user or group of users on each file system. You may want to set quotas on file systems that contain home directories, such as `/usr/users`, because the sizes of these file systems can increase more significantly than other file systems. You should avoid setting quotas on the `/tmp` file system.

7.7.3.1 Hard and Soft Quota Limits

File systems can have both soft and hard quota limits. When a hard limit is reached, no more disk space allocations or file creations that would exceed the limit are allowed. The soft limit may be reached for a period of time (called the grace period). If the soft limit is reached for an amount of time that exceeds the grace period, no more disk space allocations or file creations are allowed until enough disk space is freed or enough files are deleted to bring the disk space usage or number of files below the soft limit.

Caution

With both hard and soft limits, you can end up with a partially-written file if the quota limit is reached while you are writing to the file.

If you are in an editor and exceed a quota limit, do not abort the editor or write the file because data may be lost. Instead, use the editor exclamation point (`!`) shell escape command to remove files. You can also write the file to another file system, such as `/tmp`, remove files from the file system whose quota you reached, and then move the file back to that file system.

7.7.3.2 Activating File System Quotas

To activate quotas on a UNIX file system, perform the following steps.

1. Configure the system to include the disk quota subsystem by editing the `/sys/conf/`*`NAME`* system configuration file to include the following line:

```
options         QUOTA
```

2. Edit the /etc/fstab file and change the fourth field of the file system's entry to read rw, userquota, and groupquota.
3. Use the quotacheck command to create a quota file where the quota subsystem stores current allocations and quota limits. Refer to the quotacheck(8) reference page for command information.
4. Use the edquota command to activate the quota editor and create a quota entry for each user.

   For each user or group you specify, edquota creates a temporary ASCII file that you edit with the vi editor. Edit the file to include entries for each file system with quotas enforced, the soft and hard limits for blocks and inodes (or files), and the grace period.

   If you specify more than one user name or group name in the edquota command line, the edits will affect each user or group. You can also use prototypes that allow you to quickly set up quotas for groups of users. Refer to the edquota(8) reference page for more information.
5. Use the quotaon command to activate the quota system. Refer to the quotaon(8) reference page for more information.
6. To check and enable disk quotas during system startup, use the following command to set the disk quota configuration variable in the /etc/rc.config file:

   ```
   # /usr/sbin/rcmgr set QUOTA_CONFIG yes
   ```

If you want to turn off quotas, use the quotaoff command. Also, the umount command turns off quotas before it unmounts a file system. Refer to the quotaoff(8) reference page for more information.

7.7.4 Verifying Disk Quotas

If you are enforcing user disk quotas, you should periodically verify your quota system. You can use the quotacheck, quota, and repquota commands to compare the established limits with actual use.

The quotacheck command verifies that the actual block use is consistent with established limits. You should run the quotacheck command twice: when quotas are first enabled on a file system and after each reboot. The command gives more accurate information when there is no activity on the system.

The quota command displays the actual block use for each user in a file system. Only the root user can execute the quota command.

The `repquota` command displays the actual disk use and quotas for the specified file system. For each user, the current number of files and the amount of space (in KB) is displayed along with any quotas.

If you find it necessary to change the established quotas, use the `edquota` command, which allows you to set or change the limits for each user.

Refer to the `quotacheck`(8), `quota`(8), and `repquota`(8) reference pages for more information on disk quotas.

7.8 Partitioning Disks

This section provides the information you need to change the partition scheme of your disks. In general, you allocate disk space during the initial installation or when adding disks to your configuration. Usually, you do not have to alter partitions; however, there are cases when it is necessary to change the partitions on your disks to accommodate changes and to improve system performance.

The disk label provides detailed information about the geometry of the disk and the partitions into which the disk is divided. You can change the label with the `disklabel` command. You must be the root user to use the `disklabel` command.

There are two copies of a disk label, one located on the disk and one located in system memory. Because it is faster to access system memory than to perform I/O, when the system boots, it copies the disk label into memory. Use the `disklabel –r` command to directly access the label on the disk instead of going through the in-memory label.

Note

Before you change disk partitions, back up all the file systems if there is any data on the disk. Changing a partition overwrites the data on the old file system, destroying the data. Refer to Section 7.1.1 for more information on disk partitions.

The following rules apply to changing partitions:

- You cannot change the offset, which is the beginning sector, or shrink any partition on a mounted file system or on a file system that has an open file descriptor.
- If you need only one partition on the entire disk, use partition `c`.
- Specify the raw device for partition `a`, which begins at the start of the disk sector (sector 0), when you change the label.

Before changing the size of a disk partition, review the current partition setup by viewing the disk label. The `disklabel` command allows you to view the partition sizes. The bottom, top, and size of the partitions are in 512-byte sectors.

To review the current disk partition setup, use the following `disklabel` command syntax:

disklabel -r *device*

Specify the device with its directory name (`/dev`) followed by the raw device name, drive number, and partition `a` or `c`. You can also specify the disk unit and number, such as `rz1`.

An example of using the `disklabel` command to view a disk label follows:

```
# disklabel –r /dev/rrz3a
type: SCSI
disk: rz26
label:
flags:
bytes/sector: 512
sectors/track: 57
tracks/cylinder: 14
sectors/cylinder: 798
cylinders: 2570
rpm: 3600
interleave: 1
trackskew: 0
cylinderskew: 0
headswitch: 0           # milliseconds
track-to-track seek: 0  # milliseconds
drivedata: 0

8 partitions:
#       size offset   fstype [fsize bsize cpg]
 a:  131072       0   4.2BSD   1024  8192  16  # (Cyl.    0 - 164*)
 b:  262144  131072   unused   1024  8192      # (Cyl.  164*- 492*)
 c: 2050860       0   unused   1024  8192      # (Cyl.    0 - 2569)
 d:  552548  393216   unused   1024  8192      # (Cyl.  492*- 1185*)
 e:  552548  945764   unused   1024  8192      # (Cyl. 1185*- 1877*)
 f:  552548 1498312   unused   1024  8192      # (Cyl. 1877*- 2569*)
 g:  819200  393216   unused   1024  8192      # (Cyl.  492*- 1519*)
 h:  838444 1212416   4.2BSD   1024  8192 16   # (Cyl. 1519*- 2569*)
```

You must be careful when you change partitions because you can overwrite data on the file systems or make the system inefficient. If the partition label becomes corrupted while you are changing the partition sizes, you can return to the default partition label by using the `disklabel` command with the –`w` option, as follows:

```
# disklabel -r -w /dev/rrz1a rz26
```

The disklabel command allows you to change the partition label of an individual disk without rebuilding the kernel and rebooting the system. Use the following procedure:

1. Display disk space information about the file systems by using the df command.
2. View the /etc/fstab file to determine if any file systems are being used as swap space.
3. Examine the disk's label by using the disklabel command with the –r option. Refer to the rz(7) and ra(7) reference pages and to the /etc/disktab file for information on the default disk partitions.
4. Unmount the file systems on the disk whose label you want to change.
5. Calculate the new partition parameters. You can increase or decrease the size of a partition. You can also cause partitions to overlap.
6. Edit the disk label by using the disklabel command with the –e option to change the partition parameters, as follows:

   disklabel -e [-r] *disk*

   An editor, either the vi editor or that specified by the EDITOR environment variable, is invoked so you can edit the disk label, which is in the format displayed with the disklabel –r command.

   The –r option writes the label directly to the disk and updates the system's in-memory copy, if possible. The *disk* parameter specifies the unmounted disk (for example, rz0 or /dev/rrz0a).

   After you quit the editor and save the changes, the following prompt is displayed:

   ```
   write new label? [?]:
   ```

   Enter y to write the new label or n to discard the changes.
7. Use the disklabel command with the –r option to view the new disk label.
8. Edit the /etc/fstab file to include the new file systems.

7.9 Cloning Disks

You can use the dd command to clone a complete disk or a disk partition; that is, you can produce a physical copy of the data on the disk or disk partition.

Note

Because the dd command was not meant for copying multiple files, you should clone a disk or a partition only on a disk that is

used as a data disk or one that does not contain a file system. Use the `dump` and `restore` commands described in Chapter 12 to clone disks or partitions that contain a file system.

Digital UNIX protects the first block of a disk with a valid disk label because this is where the disk label is stored. As a result, if you clone a partition to a partition on a target disk that contains a valid disk label, you must decide whether you want to keep the existing disk label on that target disk.

If you want to maintain the disk label on the target disk, use the `dd` command with the `skip` and `seek` options to move past the protected disk label area on the target disk. Note that the target disk must be the same size as or larger than the original disk.

To determine if the target disk has a label, use the following `disklabel` command syntax:

disklabel -r *target_device*

You must specify the target device directory name (`/dev`) followed by the raw device name, drive number, and partition `c`. If the disk does not contain a label, the following message is displayed:

```
Bad pack magic number (label is damaged, or pack is unlabeled)
```

The following example shows a disk that already contains a label:

```
# disklabel -r /dev/rrz1c

type: SCSI
disk: rz26
label:
flags:
bytes/sector: 512
sectors/track: 57
tracks/cylinder: 14
sectors/cylinder: 798
cylinders: 2570
rpm: 3600
interleave: 1
trackskew: 0
cylinderskew: 0
headswitch: 0           # milliseconds
track-to-track seek: 0  # milliseconds
drivedata: 0

8 partitions:
#      size  offset  fstype [fsize bsize  cpg]
 a:  131072       0  unused 1024 8192 # (Cyl.    0 - 164*)
 b:  262144  131072  unused 1024 8192 # (Cyl.  164*- 492*)
 c: 2050860       0  unused 1024 8192 # (Cyl.    0 - 2569)
 d:  552548  393216  unused 1024 8192 # (Cyl.  492*- 1185*)
 e:  552548  945764  unused 1024 8192 # (Cyl. 1185*- 1877*)
 f:  552548 1498312  unused 1024 8192 # (Cyl. 1877*- 2569*)
```

```
 g:  819200  393216  unused 1024 8192 # (Cyl.  492*- 1519*)
 h:  838444 1212416  unused 1024 8192 # (Cyl. 1519*- 2569*)
```

If the target disk already contains a label and you do not want to keep the label, you must zero (clear) the label by using the `disklabel –z` command. For example:

```
# disklabel -z /dev/rrz1c
```

To clone the original disk to the target disk and keep the target disk label, use the following `dd` command syntax:

dd if=*original_disk* of=*target_disk* skip=16 seek=16 bs=512

Specify the device directory name (`/dev`) followed by the raw device name, drive number, and the original and target disk partitions. For example:

```
# dd if=/dev/rrz0c of=/dev/rrz1c skip=16 seek=16 bs=512
```

7.10 Checking for Overlapping Partitions

Commands to mount or create file systems, add a new swap device, and add disks to the Logical Storage Manager first check whether the disk partition specified in the command already contains valid data, and whether it overlaps with a partition that is already marked for use. The `fstype` field of the disk label is used to determine when a partition or an overlapping partition is in use.

If the partition is not in use, the command continues to execute. In addition to mounting or creating file systems, commands like `mount`, `newfs`, `fsck`, `voldisk`, `mkfdmn`, `rmfdmn`, and `swapon` also modify the disk label, so that the `fstype` field specifies how the partition is being used. For example, when you add a disk partition to an AdvFS domain, the `fstype` field is set to `AdvFS`.

If the partition is not available, these commands return an error message and ask if you want to continue, as shown in the following example:

```
# newfs /dev/rrz8c
WARNING: disklabel reports that rz8c currently
is being used as "4.2BSD" data. Do you want to
continue with the operation and possibly destroy
existing data? (y/n) [n]
```

Applications, as well as operating system commands, can modify the `fstype` of the disk label, to indicate that a partition is in use. See the `check_usage`(3) and `set_usage`(3) reference pages for more information.

8

Administering the POLYCENTER Advanced File System

This chapter introduces the POLYCENTER Advanced File System (AdvFS), which is a file system option on the Digital UNIX operating system. AdvFS provides rapid crash recovery, high performance, and a flexible structure that enables you to manage your file system while it is on line.

An optional set of utilities is available for AdvFS that expands the capabilities of the file system. The POLYCENTER Advanced File System Utilities (AdvFS Utilities) provide functions such as adding volumes without reconfiguring the directory hierarchy of the file system, cloning filesets to enable online backups, and improving system performance with file defragmentation, domain balancing, and file striping. The AdvFS Utilities also include a graphical user interface (GUI) that simplifies file system management.

The AdvFS component is licensed with the Digital UNIX operating system and is available as an optional subset during an advanced installation. You can choose AdvFS as the file system type for the root, `/usr`, or `/var` file systems. See the *Installation Guide* for more information about performing advanced installations. The AdvFS Utilities are available as a separately licensed layered product.

Before setting up AdvFS, you need to understand how it differs from traditional UNIX file systems. These differences, although minor with regard to your transition from a UNIX File System (UFS), play a role in how you plan and maintain AdvFS. For information on UFS, see Chapter 7.

The remaining sections in this chapter describe the unique characteristics of the file system design, instructions on setting up a new file system, and a clarification of what you can accomplish without the optional file system utilities.

Table 8–1 lists and describes the commands unique to the base portion of AdvFS.

Table 8–1: Advanced File System Commands

Command	Description
advfsstat	Displays AdvFS performance statistics
advscan	Locates AdvFS partitions on disks
chfile	Changes the attributes of a file
chfsets	Changes the attributes of a fileset
chvol	Changes the attributes of a volume
mkfdmn	Creates a file domain
mkfset	Creates a fileset within a file domain
renamefset	Renames an existing fileset
rmfdmn	Removes a file domain
rmfset	Removes a fileset from a file domain
shblk	Shows AdvFS blocks
shfragbf	Shows AdvFS frag file
showfdmn	Displays the attributes of a file domain
showfile	Displays the attributes of a file
showfsets	Displays the attributes of filesets in a file domain
vbmtchain	Displays bitmap metadata table (BMT) information
vbmtpg	Displays a formatted page of the bitfile metadata table (BMT)
vdump	Performs incremental backups
vfile	Displays a file
vfragpg	Displays a page of the frag file
vlogpg	Displays a page of the log file
vlsnpg	Displays the logical sequence number (LSN) of a log page
vrestore	Restores files from devices written with the vdump command
vtagpg	Displays a formatted page of the tag directory
vverify	Checks the AdvFS on-disk metadata structures

AdvFS uses the the standard UFS quota commands to establish and manage AdvFS user and group quotas. Table 8–2 lists and describes the user and group quota commands to use with AdvFS.

Table 8–2: Advanced File System Quota Commands

Command	Description
edquota	Edits user and group quotas
ncheck	Prints the tag and full pathname for each file in the file system
quot	Summarizes fileset ownership
quota	Displays disk usage and limits by user or group
quotacheck	Checks filesystem quota consistency
quotaon	Turns quotas on
quotaoff	Turns quotas off
repquota	Summarizes quotas for a file system

Table 8–3 lists and describes the optional Advanced File System Utilities that are available as a separately licensed layered product.

Table 8–3: Optional POLYCENTER Advanced File System Utilities

Utility	Description
addvol	Adds volumes to an existing file domain
balance	Balances the percentage of used space between two volumes
clonefset	Creates a read-only fileset, which you use to perform online backups
defragment	Makes the files in a domain more contiguous
dtadvfs	Starts the AdvFS graphical user interface
migrate	Moves the location of a file within a file domain
mktrashcan	Attaches directories to a trashcan directory, which stores deleted files
rmtrashcan	Detaches a specified directory from a trashcan directory
rmvol	Removes a volume from an existing file domain
shtrashcan	Shows the trashcan directory, if any, that is attached to a specified directory
stripe	Stripes a file across several volumes in a file domain

8.1 Features and Benefits

The AdvFS and AdvFS Utilities provide an innovative design that is not based on any existing file system, such as the BSD or the System V file systems. AdvFS enables you to:

- Introduce flexibility

  The new design overcomes many unexpected events, such as a sudden system interruption or running out of disk space.

  System administrators can manage disk capacity independently of the logical file system directory structure and transparently to the user. As a result, configuration planning is less complicated and more flexible. A system administrator can add or remove disks while the file system is active.

  End users can retrieve their own unintentionally deleted files from predefined trashcan directories.

- Maintain compatibility

  Logical structures, quota controls, and backup capabilities are familiar, which promotes a smooth transition to the new file system.

  From a user's perspective, AdvFS behaves like any UNIX file system. End users can use the `mkdir` command to create new directories, the `cd` command to change directories, and the `ls` command to list directory contents. Application programmers encounter a programming interface based on UNIX: `open()`, `close()`, `creat()`, and so on. AdvFS is POSIX 1003.1 compliant.

  AdvFS replaces or eliminates several standard commands, such as `newfs`, `dump`, `restore`, and `fsck`.

- Improve data availability

  Rebooting after a system interruption is fast. AdvFS uses write-ahead logging, instead of the `fsck` utility, as a way to check for and repair file system inconsistencies that can occur during an unexpected system crash or power failure. The number of uncommitted records in the log, not the amount of data in the file system, dictates the speed of a recovery. As a result, reboots are quick and predictable, measured in seconds instead of minutes. User file data can be logged as well, providing better control over data recovery.

  To maintain data availability, system administrators can perform backups, file system reconfiguration, and file system tuning without taking the system off line.

- Provide high performance

An extent-based file allocation scheme allows for fewer and larger I/O transfers, which increases sequential read/write throughput and simplifies large data transfers. The file system performs large reads from disk when sequential access is anticipated. It also performs large writes by combining adjacent data into a single disk transfer.

The AdvFS Utilities enable multivolume file systems. Multiple volumes allow file-level striping and file migration. File-level striping improves file transfer rates by spreading I/O among several disks. File migration allows for load and capacity balancing and reduces file fragmentation.

Table 8–4 lists AdvFS features and their benefits, which are unavailable to traditional file systems such as the BSD or the System V file systems. Note that some of the features listed require the optional POLYCENTER AdvFS Utilities license.

Table 8–4: Advanced File System Features and Benefits

Feature	Benefit
Rapid crash recovery	Write-ahead logging eliminates the requirement to use the `fsck` utility when recovering from an unexpected system failure and makes file system recovery time rapid and independent of file system size.
Unified buffer cache	This cache interacts with the virtual memory system to dynamically adjust the amount of physical memory being used to cache file data.
Extended capacity	The design supports large-scale storage systems by extending the size of both files and file systems. It is designed to handle files and filesets as large as nearly 16 terabytes.
Disk spanning	A file or file system can span multiple disks within a pool of disk volumes, which you can adjust to match your storage needs. (This feature requires the optional AdvFS Utilities.)
Graphical User Interface	Simplifies system management by organizing AdvFS functions into menu-selected tasks and graphically depicting file-system status. (This feature requires the optional AdvFS Utilities.)
Online resizing	You can dynamically change the size of the file system by adding or removing disk volumes while the system remains in use. This enables both online storage configuration and online file system maintenance. (This feature requires the optional AdvFS Utilities.)
File-level striping	This feature improves file transfer rates by distributing file data across multiple disk volumes. (This feature requires the optional AdvFS Utilities.)

Table 8–4: Advanced File System Features and Benefits (cont.)

Feature	Benefit
On-line performance tuning	System performance can be tuned using an array of utilities and can be performed without interrupting system users.
Online backup	You can back up the contents of your file system to media without interrupting the workflow of system users.
File undelete	System users can recover unintentionally deleted files, thus improving data availability. (This feature requires the optional AdvFS Utilities.)

8.2 AdvFS Design Overview

Unlike UFS, AdvFS consists of two distinct layers; each layer contains different file system mechanisms. The directory hierarchy layer implements the file-naming scheme and Digital UNIX file system-compliant functions such as creating and opening files or reading and writing to files. The physical storage layer implements write-ahead logging, caching, file allocation, and physical disk I/O functions.

The decoupled file system structure enables you to manage the physical storage layer apart from the directory hierarchy layer. This means that you can move files between a defined group of disks, without changing file pathnames. Since the pathnames remain the same, the action is completely transparent to end users.

The two-layer structure is the cornerstone of AdvFS. Supporting the design are two file system concepts: the file domain and the fileset.

8.2.1 File Domains

The following sections describe file domains and filesets in more detail. A file domain is a named set of one or more volumes that provides a shared pool of physical storage. With respect to file domains, a volume is any mechanism that behaves like a UNIX block device: an entire magnetic disk, a disk partition, or a logical volume that is configured with the Logical Volume Manager (LVM) or the Logical Storage Manager (LSM).

Creating a file domain is the first step in setting up AdvFS. The number of file domains that you construct on your system depends on the needs of your site, the resources available to the system, and the number of independent file systems you choose to manage.

When created, all file domains consist of a single (or initial) volume. You transform a single-volume file domain into a multiple-volume file domain by adding one or more volumes. Unless you have installed the optional file system utilities and registered the license PAK, you are limited to creating single-volume file domains.

The `/etc/fdmns` directory, which the file system automatically creates and updates for you, contains a subdirectory for each file domain on your system. The subdirectories contain a symbolic link to every volume in the file domain.

Use the following guidelines for file domains:

- A volume is anything that behaves like a UNIX block device. You can avoid I/O scheduling contention and enhance system performance by dedicating the entire disk (generally partition `c`) to a file domain.
- You can have 100 active file domains per system. A file domain is active when at least one of its filesets is mounted.
- The number of file domains per system depends on the needs of your site and the resources available to the system. Combining multiple volumes within a single file domain reduces the overall management cost.
- Although the risk of media failure is slight, a single failure within a file Domain renders the entire file domain useless. In the case of media failure, you must re-create the file domain and restore all the files. Thus, the more volumes that you add to your file domain, the greater the risk that a domain will fail.

  To reduce the risk of domain failure, limit the number of volumes to 3 volumes per file domain.
- To maintain high performance, avoid splitting a disk between two file domains. For example, do not add partition `g` to one file domain and partition `h` of the same disk to another file domain.

Refer to the `addvol`(8), `mkfdmn`(8), `advfs`(4), `fdmns`(4), and `showdmn`(8) reference pages for more information.

8.2.2 Filesets and File Systems

A fileset represents a portion of the directory hierarchy of a file system; each fileset, which has a unique name, is a collection of directories and files that form a subtree structure. Because the hierarchy layer is independent of the storage layer, you can manage file placement without affecting the logical structure of filesets.

Filesets and file systems are equivalent in many ways. For instance, you mount filesets individually like you mount file systems. Similarly, filesets are units on which you enable quotas and back up data.

Although you can set up a fileset to simulate a traditional file system, filesets offer more flexibility. For instance, you can create multiple filesets in place of one file system, allowing you to manage each fileset independently. Conversely, you can create one large fileset in a situation where you usually define multiple file systems, thereby reducing management overhead.

Unlike file systems, filesets can have clone filesets. A clone fileset is a read-only copy of an existing fileset that you create to capture your data at one moment in time. You can back up the contents of the clone fileset to media while the original fileset remains available to system users. The clone fileset utility, `clonefset`, is available with the POLYCENTER Advanced File System Utilities.

Use the following guidelines for filesets:

- You can create an unlimited number of filesets per system; however, the number of filesets that you can mount simultaneously is limited to 512 minus the number of active file domains.
- The correct number of filesets per domain depends on your requirements. You manage filesets independently of each other (mount, backup, and quotas). The more filesets that you establish, the greater your flexibility. On the other hand, a greater number of filesets increases management effort.

Refer to the `advfs`(4), `fdmns`(4), `mkfset`(8), and `showfsets`(8) reference pages for more information.

8.3 File Storage Allocation

The Advanced File System (AdvFS) always attempts to write each file to disk as a set of contiguous units called pages; a page is 8 KB of disk space. Contiguous, in this context, means storage on disk that is physically adjacent. A set of one or more contiguous pages is called an extent. Contiguous placement of the pages means that the I/O mechanism works more efficiently. When a file consists of many small extents, the I/O mechanism must work harder to read or write that file.

8.3.1 Allocation Policy

Files are not static; disk space requirements change over time. To maintain contiguous file placement without over-allocating space on the disk, AdvFS applies a policy to file storage allocation. Each time a file is appended, AdvFS adds pages to the file by preallocating one-fourth of the file size up to 16 pages. If a large write requires more file space, AdvFS attempts to allocate up to 256 contiguous pages. Excess preallocated space is truncated

when the file is closed. Unused preallocated space is then available for the next write.

For multivolume file domains, new files are allocated sequentially across volumes. Volumes that are more than 86% full (allocated) are not used for new file allocation unless all volumes are over 86% full. When existing files are appended, storage is allocated on the volume on which the file was initially allocated, until that volume is full.

When a new volume is added to a file domain, it is added to the storage allocation sequence. Files are allocated to the new volume in turn.

8.3.2 Fragments

AdvFS writes files to disk in sets of 8-KB pages. When a file uses only part of the last page, less than 8 KB, a file *fragment* is created. The fragment, which is from 1 KB to 7 KB in size, is allocated from the *frag file*. Using fragments considerably reduces the amount of unused, wasted disk space. Note that the frag file is a special file not visible in the directory hierarchy.

8.3.3 Policy Allocation Limitations

Given the dynamic nature of a file system, the file storage-allocation policy cannot always guarantee contiguous page placement. The following factors affect the policy:

- Excessive disk fragmentation

  When a disk is fragmented, AdvFS writes data to isolated physical pages, based on availability, instead of writing to contiguous pages.

- Multiple users

  Many users on a system increases file activity. As a result, files become more fragmented on the disk.

File fragmentation can reduce the I/O performance of AdvFS. You can use the `defragment` utility to reduce domain fragmentation. The `defragment` utility is available with the optional AdvFS Utilities. See the `defragment`(8) reference page for information on reducing file fragmentation.

8.4 Setting Up the Advanced File System

As you begin planning, decide whether you want to set up AdvFS to resemble a traditional UFS configuration. Once you become familiar with AdvFS, you can begin to move away from the traditional model.

When planning your configuration, consider setting up the root and /usr file systems on AdvFS. Using AdvFS as the root file system enables booting from an AdvFS file domain and makes AdvFS features available on all local file systems. By having the /usr file system on AdvFS you can significantly reduce the amount of time your system is down after a system failure.

You can put the root file system and /usr file system on AdvFS during the initial base system installation. If you prefer, you can convert your existing root and /usr file systems after installation. See Section 8.9 and Section 8.10 for conversion guidelines.

The following procedure is a guide for setting up an active, single-volume file system:

1. Create a single-volume file domain by using the mkfdmn command.

   You can add more volumes to any existing file domain (except for the root_domain) by using the addvol utility, if you have installed the file system utilities.

2. Create one or more filesets by using the mkfset command.

   Name each fileset the same as its mount-point directory; for example, if the mount-point directory is /tmp, name the fileset tmp.

3. Create the mount-point directory by using the mkdir command.

4. Mount each fileset by using the mount command.

In the unlikely event of a severe failure where you must restore the /fdmns directory manually by reconstructing that directory, you must have a separate record of your file system configuration with the name of each file domain and its associated volumes. Always keep this record up-to-date.

The following examples use AdvFS to set up active file systems. The file-domain configurations in the examples are:

- /usr file system in domain1 on /dev/rz3c

  This example mounts one fileset, called usr, on the /usr mount-point directory, which already exists on the system:

  ```
  # mkfdmn /dev/rz3c domain1
  # mkfset domain1 usr
  # mount -t advfs domain1#usr /usr
  ```

- /tmp and /public file systems in domain2 on /dev/rz2c

  This example creates a single volume domain, domain2, and two filesets in the domain, tmp and public. Because the domain has only one volume, the files in both filesets reside on one volume. This is an accepted configuration on AdvFS.

To set up this configuration, the example mounts two filesets, called `tmp` and `public`, on the respective mount-point directories:

```
# mkfdmn /dev/rz2c domain2
# mkfset domain2 tmp
# mkfset domain2 public
# mkdir /public
# mount -t advfs domain2#tmp /tmp
# mount -t advfs domain2#public /public
```

- `/projects` file system on `rz1c` (`domain3`)

  This example mounts one fileset, called `projects`, on the `/projects` mount-point directory, which the example creates:

```
# mkfdmn /dev/rz1c domain3
# mkfset domain3 projects
# mkdir /projects
# mount -t advfs domain3#projects /projects
```

The number sign (#) between the file domain and fileset is part of the syntax that represents a fileset; this character does not indicate a comment.

Refer to the `mkfdmn`(8), `mkfset`(8), and `addvol`(8) reference pages for more information.

8.5 Managing File System and Fileset Quotas

AdvFS eliminates the slow reboot activities associated with quotas on UFS. As a result, enabling quotas is a useful way of tracking and controlling the amount of physical storage that each fileset consumes.

The AdvFS quota system is compatible with the Berkeley-style quotas of UFS. Basically, AdvFS supports user account and group quotas. However, the AdvFS quota system differs in two ways: AdvFS differentiates between quota maintenance and quota enforcement and supports fileset quotas.

- Quota maintenance

  AdvFS quota maintenance tracks file and disk space usage. It keeps a record of the number of files and blocks a user or group is using in the `quota.user` and `quota.group` files found in the root directory of a fileset. The AdvFS quota system always maintains quota information. Unlike UFS, this function cannot be disabled.

- Quota enforcement

  When quota enforcement is enabled, the AdvFS quota system enforces all quota limits set by the system administrator. You use the `edquota` command to set quota limits. Use the `quotaon` and `quotaoff` commands to enable and disable quota enforcement.

The AdvFS user and group quota commands are the same as UFS quota commands. Table 8–5 lists the quota commands used to set and maintain user and group quotas.

Table 8–5: Advanced File System Quota Commands

Command	Description
`edquota`	Edits user and group quotas.
`ncheck`	Prints a list of pairs (tag and pathname) for all files in a specified fileset. Use the sorted output as input for the `quot` command.
`quot`	Prints the number of blocks in the named fileset currently owned by each user.
`quota`	Displays users' disk usage and limits on filesets that have quotas enabled.
`quotacheck`	Checks file system quota consistency.
`quotaon`	Enables quotas on one or more filesets. By default, quotas are enabled on all filesets.
`quotaoff`	Disables quotas on one or more filesets.
`repquota`	Prints a summary of the disk usage and quotas for the specified file systems.

The quota commands display disk usage in block sizes of 1024-byte blocks.

- Fileset quotas

  In addition to the Berkeley-style user and group quotas, AdvFS also supports fileset quotas. Fileset quotas are similar to user/group quotas in both function and management.

  Fileset quotas apply to the fileset rather than individual users and/or groups. You can use them to limit the amount of disk storage and number of files consumed by a fileset. This is useful when a file domain contains several filesets. Without fileset quotas, all filesets have access to all disk space in a file domain, allowing one fileset to use all the disk space in a file domain.

  Filesets can have both soft and hard disk storage and file limits. When a hard limit is reached, no more disk space allocations or file creations that would exceed the limit are allowed. The soft limit may be exceeded for a period of time (called the grace period). If the soft limit is exceeded for an amount of time that exceeds the grace period, no more disk space allocations or file creations are allowed until enough disk space is freed or enough files are deleted to bring the disk space usage or number of files below the soft limit. The grace periods for the soft limits are set

with the edquota -tg command (the -g switch must be used because AdvFS uses the group quota file of the fileset to maintain fileset quotas). Fileset quotas are managed using the commands shown in Table 8–6.

Table 8–6: Fileset Quota Commands

Command	Description
chfsets	Changes file usage and block usage limits (quotas) for a fileset
df	Displays the limits and actual number of blocks used by a fileset
edquota	Sets the grace period for fileset quotas
showfsets	Displays the file and block usage limits for the filesets in a domain
showfdmn	Displays space usage for the specified domain

Refer to the chfsets(8,) showfsets(8,) showfdmn(8,) edquota(8), ncheck(8), quot(8), quota(1), quotacheck(8), quotaon(8), and vrepquota(8) reference pages for more information.

8.6 Backing Up Data

The dump command supports UFS exclusively. As a result, AdvFS provides an equivalent backup command called vdump. AdvFS also replaces the restore command with the equivalent vrestore command.

If you already use the dump or restore commands to back up and restore data, then the vdump and vrestore commands will be familiar to you.

There are differences between the UFS commands and AdvFS commands. The vdump command supports other file system types, so you can use vdump command capabilities on files systems other than AdvFS. Several minor flags are absent from AdvFS commands; new flags increase your access to information when backing up and restoring files. The vdump command also provides the following features that are unavailable with the dump command:

- Extended support

  This feature expands the support of the vdump command to other file systems, including UFS. This means that you can simplify the task of backing up multiple file systems by using the same backup facility across the system. Other file systems can benefit from the extended features provided by the vdump command.

- Subdirectory backups

This feature refines the granularity of your system backups. Instead of backing up an entire file system, you can selectively back up individual subdirectories by using the `-D` flag.

- Data compression

  This feature writes data in a compressed form to a saveset, which reduces storage usage and runs faster on slow backup devices by writing less data.

Table 8–7 lists and describes new command flags for the `vdump` command.

Table 8–7: The vdump Command Flags

Flag	Description
`-C`	Compresses data during a backup
`-D`	Backs up a subdirectory
`-F`	Specifies the number of in-memory buffers
`-V`	Displays the current command version number
`-h`	Displays usage help
`-q`	Displays error messages, but not warning messages
`-v`	Displays the names of files as they are backed up
`-x`	Increases saveset error protection

Table 8–8 lists and describes new command flags for the `vrestore` command.

Table 8–8: The vrestore Command Flags

Flag	Description
`-V`	Displays the current command version number
`-l`	Lists the saveset structure
`-q`	Displays error messages, but not warning messages
`-o`	Provides file overwrite options

Refer to the `vdump`(8) and `vrestore`(8) reference pages for more information.

8.7 Restoring the fdmns Directory

The `/etc/fdmns` directory contains a set of subdirectories, one for each file domain on your system. Each subdirectory includes symbolic links to every

volume in the file domain. AdvFS cannot mount filesets without this directory.

Note

You must use the `addvol` and `rmvol` utilities to add and remove volumes. Creating and removing links alone does not add or remove a volume.

AdvFS creates a corresponding subdirectory each time you create a file domain. For example, you can create a file domain called `mydomain`, which contains the volume `/dev/rz1c`. The file system creates the `/etc/fdmns/mydomain` subdirectory, which contains a symbolic link to `/dev/rz1c`. When you add or remove a volume from the file domain, the file system updates the subdirectory by adding or removing symbolic links.

In some ways, the `/etc/fdmns` directory resembles the `/etc/fstab` file; each has special significance and requires extra attention. You must restore the `/etc/fdmns` directory if its contents are deleted, corrupted, or if you install a new version of the operating system. Although a missing or damaged `/etc/fdmns` directory prevents access to the file domain, the data within the file domain remains intact.

8.7.1 Restoring from Backup Media

Restoring from backup media is the preferable method for restoring the `/etc/fdmns` directory, provided you have a current backup copy of the directory. You can use any standard backup facility (`vdump`, `dump`, `tar`, or `cpio`) to back up the `/etc/fdmns` directory. To restore the directory, use a recovery procedure that is compatible with your backup facility.

Always back up the `/etc/fdmns` directory whenever you create a new file domain, add a volume to an existing file domain, or remove a volume from an existing file domain.

8.7.2 Reconstructing the Directory

You can reconstruct the `/etc/fdmns` directory manually or with the `advscan` command. The procedure for reconstructing the `fdmns` directory is similar for both single-volume and multivolume file domains.

If you choose to reconstruct manually, you must know the name of each file domain on your system and its associated volumes. In other words, you need detailed records of the file domains on your system.

The following example manually reconstructs two file domains, each containing a single volume (or special device). The file domains are:

- domain1, on /dev/rz1c
- domain2, on /dev/rz2c

To reconstruct the two single-volume file domains, enter:

```
# mkdir /etc/fdmns
# mkdir /etc/fdmns/domain1
# cd /etc/fdmns/domain1
# ln -s /dev/rz1c
# mkdir /etc/fdmns/domain2
# cd /etc/fdmns/domain2
# ln -s /dev/rz2c
```

The following example, which requires that the optional AdvFS Utilities be installed, reconstructs one multivolume file domain. The domain1 file domain contains three volumes:

- /dev/rz1c
- /dev/rz2c
- /dev/rz3c

To reconstruct the multivolume file domain, enter:

```
# mkdir /etc/fdmns
# mkdir /etc/fdmns/domain1
# cd /etc/fdmns/domain1
# ln -s /dev/rz1c
# ln -s /dev/rz2c
# ln -s /dev/rz3c
```

Refer to the fdmns(4), mkfdmn(8), and addvol(8) reference pages for more information.

You can use the advscan command to rebuild all or a part of your /etc/fdmns file domain. The advscan command can perform the following tasks:

1. List partitions in the order they are found on disk.
2. Scan all disks found in any file domain.
3. Re-create missing domains.
4. Fix the domain count and links if you specify a domain.
5. Includes Logical Storage Manager (LSM) disk groups.

See the advscan(8) reference page for details on using the command to restore the /etc/fdmns directory.

8.8 Restarting the System

Unexpected shutdowns, usually as a result of system interruption or media failure, cause you to restart your Digital UNIX operating system. When you are forced to restart your system after an unexpected shutdown, AdvFS is affected.

8.8.1 System Interruption

An example of a system interruption is when your site unexpectedly loses power. This usually happens without warning.

AdvFS uses write-ahead logging as a way to reduce the impact of system interruptions. As your system reboots, the file system scans all records in the recovery log. Any operations that were uncommitted when the interruption occurred are undone. Thus, the number of uncommitted records in the log determines the speed of the recovery. Since the recovery depends on the number of records in the log, instead of the amount of data in the file system, the recovery process dramatically improves. The default log size is 4 MB.

AdvFS automatically initiates crash recovery on a file domain as soon as you mount a fileset within that file domain. You can add filesets to the `/etc/fstab` file (at least one fileset per domain) so that all file domains recover during the system reboot.

8.8.2 Media Failure

Newer magnetic disks fail less frequently than devices based on older technology. Nevertheless, if any single disk in a file domain fails, you must restore all filesets in the file domain. Assuming you use the `vdump` command to back up your filesets, you can restore your filesets by using the `vrestore` command.

Refer to the `advfs`(4), `fdmns`(4), `vdump`(8), and `vrestore`(8) reference pages for more information.

8.9 Converting the root File System

Converting the root file system to AdvFS enables booting from an AdvFS file domain and supports AdvFS as the root file system. The AdvFS root domain must reside on a single disk.

This section presents instructions for converting the root file system from UFS to AdvFS. These instructions are guidelines, that is, suggestions to

illustrate the process of converting the root file system to AdvFS. Specific file names and disk partitions can vary, depending on your system.

You can convert the UFS root file system on one disk to the equivalent AdvFS root file system on a different target disk.

Requirements:

- Root-user privilege
- A second bootable disk (you must use partition `a` or `c`)
- AdvFS installed on your system

Assumptions:

- Existing UFS configuration

  File system: `root`
  Mount directory: `/`
  Disk partition: `/dev/rz1a`

- New AdvFS configuration

  File system: `root`
  Mount directory: `/newroot`
  Disk partition: `/dev/rz2a`
  File domain: `root_domain`
  Fileset: `root`

Use the following procedure as a guide for converting the root file system:

1. Log in as root on the system containing the root file system.
2. Create a file domain and fileset.

   ```
   # mkfdmn -r -t rz26 /dev/rz2a root_domain
   # mkfset root_domain root
   ```

3. Create a mount-point directory and mount the new fileset on the directory.

   ```
   # mkdir /newroot
   # mount –t advfs root_domain#root /newroot
   ```

4. Restore the UFS root file system to the root fileset.

   ```
   # vdump 0f - / | (cd /newroot; vrestore -xf -)
   ```

5. Make the disk with the root domain a bootable disk.

   ```
   # disklabel -r /dev/rrz2a > /tmp/rz2label
   # disklabel -t advfs -r -R /dev/rrz2a /tmp/rz2label rz26
   ```

6. Edit the /etc/fstab file on the AdvFS root fileset to indicate the new root entry.

   a. Search /newroot/etc/fstab for the entry that previously mounted root as a UFS file system, such as:

      ```
      /dev/rz1a    /  ufs rw 1 1
      ```

   b. Comment out this entry by preceding it with a pound sign (#).

   c. Add the following line:

      ```
      root_domain#root  /  advfs rw 1 1
      ```

7. Shut down the system and reset the boot default device, BOOTDEF_DEV to point to the disk with the new root domain.
8. Reboot the system to enable the AdvFS root file system.

The converted root file system is ready to use.

The AdvFS root domain is limited to one disk. Do not use the addvol command to extend the root domain.

8.10 Converting the /usr File System from UFS to AdvFS

Relying on the fsck utility to check and repair the /usr file system can be time-consuming. By converting the /usr (UFS) file system to AdvFS, you can reduce the amount of time your system is down after a system failure.

This section presents several methods for converting the /usr file system from UFS to AdvFS. These methods are guidelines to illustrate the process of converting file systems to AdvFS: Specific file names, tape drives, and disk partitions can vary, depending on your system.

8.10.1 Using a Backup Tape to Convert the /usr File System from UFS to AdvFS

You can convert the /usr (UFS) file system to the equivalent /usr (AdvFS) file system by backing up the existing file system to tape and restoring it to an AdvFS environment.

Requirements:

- Root-user privilege
- Backup device and media
- Five percent more disk space for the converted file system
- AdvFS installed on your system

Assumptions:

- Existing UFS configuration

  File system: /usr
  Disk partition: /dev/rz3g

- New AdvFS configuration

  File system: /usr
  Disk partition: /dev/rz3g
  File domain: usr_domain
  Fileset: usr

Use the following procedure as a guide for converting the /usr file system:

1. Log in as root on the system that contains the /usr file system.
2. Back up the /usr file system to /dev/rmt0h, the default tape drive, by entering the following sequence of commands:

   ```
   # mt rewind
   # cd /usr
   # vdump –0 .
   ```

3. Edit the /etc/fstab file.

   a. Search for the entry that mounts /usr as a UFS file system.

      ```
      /dev/rz3g          /usr       ufs rw 1 2
      ```

   b. Replace the previous line with the following entry, which mounts /usr as an AdvFS file system:

      ```
      usr_domain#usr    /usr        advfs  rw
      ```

4. Shut down the system.

   ```
   # shutdown –h now
   ```

5. Reboot the system to single-user mode.

   ```
   >>> b -fl i
   ```

   The system will prompt you for the name of the kernel you want to boot. Press Return to accept the default vmunix kernel.

6. In single-user mode, mount the root file system as rw, create the usr_domain file domain, and create the usr fileset by entering the following sequence of commands:

   ```
   # mount –u /
   # mkfdmn /dev/rz3g usr_domain
   # mkfset usr_domain usr
   ```

7. Mount the usr fileset on the /usr directory.

```
# mount -t advfs usr_domain#usr /usr
```

8. Restore the /usr file system from tape to the usr fileset.

```
# vrestore -x -D /usr
```

9. Continue booting the system to multiuser mode. Once the system prompt returns, the converted /usr file system is ready to use.

8.10.2 Using an Intermediate File to Convert from UFS to AdvFS

You can convert the /usr (UFS) file system to the equivalent /usr (AdvFS) file system by backing up the existing file system to a file and restoring it to an AdvFS environment.

Requirements:

- Root-user privilege
- Disk space for an intermediate file

  (The file system that contains the intermediate file can be on the same disk or on a different disk. However, do not put the intermediate file on the /usr file system.)
- Five percent more disk space for the converted file system
- AdvFS installed on your system

Assumptions:

- Existing UFS configuration

  File system: /usr
  Disk partition: /dev/rz3g
  Intermediate file: /tmp/usr_bck
- New AdvFS configuration

  File system: /usr
  Disk partition: /dev/rz3g
  File domain: usr_domain
  Fileset: usr

Use the following procedure as a guide for converting the /usr file system:

1. Log in as root on the system that contains the /usr file system.
2. Back up the /usr file system to /tmp/usr_bck, the intermediate file, by entering the following sequence of commands:

```
# cd /usr
# vdump -0f /tmp/usr_bck /usr
```

3. Edit the /etc/fstab file.

   a. Search for the entry that mounts /usr as a UFS file system.

```
/dev/rz3g          /usr       ufs rw 1 2
```

b. Replace the previous line with the following entry, which mounts /usr as an AdvFS file system:

```
usr_domain#usr    /usr        advfs  rw
```

4. Shut down the system:

```
# shutdown –h now
```

5. Reboot the system in single-user mode:

```
>>> b -fl i
```

The system will prompt you for the name of the kernel you want to boot. Press Return to accept the default vmunix kernel.

6. In single-user mode, mount the root file system as rw, create the usr_domain file domain, and create the usr fileset by entering the following sequence of commands:

```
# mount –u /
# mkfdmn /dev/rz3g usr_domain
# mkfset usr_domain usr
```

7. Mount the usr fileset on the /usr directory:

```
# mount –t advfs usr_domain#usr /usr
```

8. Restore the /usr file system from the intermediate file to the usr fileset:

```
# vrestore –xf /tmp/usr_bck –D /usr
```

9. Continue booting the system to multiuser mode. Once the system prompt returns, the converted /usr file system is ready to use.

8.10.3 Converting from One Disk to Another Disk

You can convert the /usr (UFS) file system on one disk to the equivalent /usr (AdvFS) file system on a different target disk.

Requirements:

- Root-user privilege
- A second disk, with 5 percent more disk space for the converted file system
- AdvFS installed on your system

Assumptions:

- Existing UFS configuration

File system: /usr
Disk partition: /dev/rz3g

- New AdvFS configuration

  File system: /usr
  Disk partition: /dev/rz2c
  Mount directory: /usr.advfs
  File domain: usr_domain
  Fileset: usr

Use the following procedure as a guide for converting the /usr file system:

1. Log in as root on the system that contains the /usr file system.
2. Create a file domain and fileset by entering the following sequence of commands:

   ```
   # mkfdmn /dev/rz2c usr_domain
   # mkfset usr_domain usr
   ```

3. Create a mount-point directory and mount the new fileset on the directory by entering the following sequence of commands:

   ```
   # mkdir /usr.advfs
   # mount -t advfs usr_domain#usr /usr.advfs
   ```

4. Change to the /usr directory.

   ```
   # cd /usr
   ```

5. While there is no activity on the system, copy the contents of the UFS file system to the AdvFS file system.

   ```
   # vdump -0f - -D . | vrestore -xf - -D /usr.advfs
   ```

6. Edit the /etc/fstab file.

   a. Search for the entry that mounts /usr as a UFS file system:

      ```
      /dev/rz3g          /usr       ufs rw 1 2
      ```

   b. Replace the previous line with the following entry, which mounts /usr as an AdvFS file system:

      ```
      usr_domain#usr     /usr       advfs  rw
      ```

7. Shut down and reboot the system. Once the system prompt returns, the converted /usr file system is ready to use.

8.11 Converting a Data File System from UFS to AdvFS

By converting your data file systems to AdvFS, you can eliminate lengthy reboots. Moreover, you can introduce new configurations to reduce file system management overhead.

This section presents two different methods for converting data file systems from UFS to AdvFS. The second method appends the first method with additional instructions, resulting in an AdvFS file system that consists of two independent filesets within one file domain.

The conversion methods presented here are only guidelines to illustrate the process of converting file systems to AdvFS. Specific file names, tape drives, and disk partitions can vary, depending on your system.

8.11.1 Using a Backup Tape to Convert a Data File System from UFS to AdvFS

You can convert a data UFS file system to the equivalent data AdvFS file system by backing up the existing file system to tape and restoring it to an AdvFS environment.

Requirements:

- Root-user privilege
- Backup device and media
- Five percent more disk space for the converted file system
- AdvFS installed on your system

Assumptions:

- Existing UFS configuration

  File system: `/staff2`
  Mount directory: `/staff2`
  Disk partition: `/dev/rz2c`

- New AdvFS configuration

  File system: `/staff2`
  Disk partition: `/dev/rz2c`
  File domain: `staff_domain`
  Fileset: `staff2`

Use the following procedure as a guide for converting the `/staff2` file system:

1. Log in as root on the system that contains the `/staff2` file system.
2. Back up the `/staff2` file system to `/dev/rmt0h`, the default tape drive, by entering the following sequence of commands:

```
# mt rewind
# cd /staff2
# vdump –0 .
```

3. Create a file domain and fileset by entering the following sequence of commands:

```
# umount /staff2
# mkfdmn /dev/rz2c staff_domain
# mkfset staff_domain staff2
```

4. Mount the new fileset on the directory by entering the following command:

```
# mount -t advfs staff_domain#staff2 /staff2
```

5. Restore the /staff2 file system from tape to the staff2 fileset.

```
# vrestore -x -D /staff2
```

6. Edit the /etc/fstab file.

   a. Search for the entry that previously mounted /staff2 as a UFS file system:

   ```
   /dev/rz2c          /staff2       ufs rw 1 2
   ```

   b. Replace the previous line with the following entry, which mounts /staff2 as an AdvFS file system:

   ```
   staff_domain#staff2     /staff2     advfs  rw
   ```

The converted /staff2 file system is ready to use.

8.11.2 Transferring an Existing Data File System and Converting It to AdvFS

You can transfer an existing data file system to a new system, then convert the file system to AdvFS.

Requirements:

- Two systems: one system that supports the tar utility and one Digital UNIX system
- Root-user privilege on the target system
- Five percent more disk space for the converted file system
- AdvFS installed on the target system

Assumptions:

- Existing UFS configuration

  File system: /staff4
- New AdvFS configuration

File system: /staff4
Disk partition: /dev/rz2c
Mount directory: /staff4
File domain: staff_domain
Fileset: staff4

Use the following procedure as a guide for converting the /staff4 file system:

1. Log in to the system that contains the /staff4 file system and back up the file system by entering the following command:

   # **tar c /staff4**

2. Log in as root on the target system.
3. Create a fileset in the staff_domain file domain.

   # **mkfset staff_domain staff4**

4. Create a mount-point directory and mount the new fileset on the directory by entering the following sequence of commands:

   # **mkdir /staff4**
   # **mount –t advfs staff_domain#staff4 /staff4**

5. Restore the /staff4 file system from /dev/rmt0h, the default tape drive, by entering the following sequence of commands:

   # **mt rewind**
   # **tar x /staff4**

6. Edit the /etc/fstab file. Add the following line, which mounts /staff4 as an AdvFS file system:

   ```
   staff_domain#staff4     /staff4     advfs  rw
   ```

7. The staff_domain file domain now includes both filesets, which are ready to use.

9

Administering the Logical Storage Manager

The Logical Storage Manager (LSM) software provides disk management capabilities that increase data availability and improve disk I/O performance. System administrators use LSM to perform disk management functions dynamically without disrupting users or applications accessing data on those disks.

LSM replaces the Logical Volume Manager (LVM) on Digital UNIX systems. Refer to the *Logical Storage Manager* manual for information about how to migrate from LVM to LSM.

9.1 Features and Benefits

Table 9–1 summarizes the LSM features and benefits.

Table 9–1: LSM Features and Benefits

Feature	Benefit
Manages disks	Frees you from the task of partitioning disks and allocating space. However, LSM allows you to keep control over disk partitioning and space allocation, if desired.
Allows transparent disk configuration changes	Allows you to change the disk configuration without rebooting or otherwise interrupting users. Also allows routine administrative tasks, such as file system backup, with reduced down time.
Stores large file systems	Enables multiple physical disks to be combined to form a single, larger logical volume. This capability, called *concatenation*, removes limitations imposed by the actual physical properties of individual disk sizes. It does this by combining the storage potential of several devices. Note that disk concatenation is available on all systems, including those that do not have an LSM software license.
Ease of system management	Simplifies the management of disk configurations by providing convenient interfaces and utilities to add, move, replace, and remove disks.

Table 9–1: LSM Features and Benefits (cont.)

Feature	Benefit
Protects against data loss	Protects against data loss due to hardware malfunction by creating a *mirror* (duplicate) image of important file systems and databases.
Increases disk performance	Improves disk I/O performance through the use of *striping*, which is the interleaving of data within the volume across several physical disks.

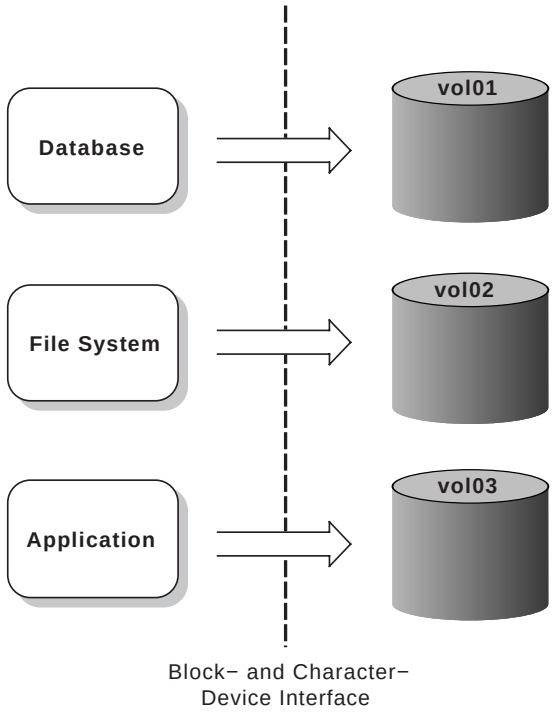
This chapter provides an overview of LSM concepts and some commonly used commands. The `volintro`(8) reference page provides a quick reference of LSM terminology and command usage. Refer to the manual *Digital UNIX Logical Storage Manager* for more complete information on LSM concepts and commands.

9.2 Understanding the LSM Components

LSM consists of physical disk devices, logical entities, and the mappings that connect the physical and logical objects.

LSM builds virtual disks, called volumes, on top of UNIX system disks. A *volume* is a special device that contains data managed by a UNIX file system, a database, or other application. LSM transparently places a volume between a physical disk and an application, which then operates on the volume rather than on the physical disk. A file system, for instance, is created on the LSM volume rather than a physical disk. Figure 9–1 shows disk storage management in an LSM configuration.

Figure 9–1: LSM Disk Storage Management



On a system that does not have LSM installed, I/O activity from the UNIX system kernel is passed through disk device drivers that control the flow of data to and from disks. When LSM is installed, the I/O passes from the kernel to the LSM volume device driver, then to the disk device drivers.

The LSM software maps the logical configuration of the system to the physical disk configuration. This is done transparently to the file systems, databases, and applications above it because LSM supports the standard block device and character device interfaces to store and retrieve data on LSM volumes. Thus, you do not have to change applications to access data on LSM volumes.

The block device special files associated with LSM volumes exist in the `/dev/vol` directory and the character device special files associated with LSM volumes exist in the `/dev/svol` directory.

9.2.1 LSM Objects

LSM logically binds together the disk devices into a volume that represents the disks as a single virtual device to applications and users. LSM uses a

structure of LSM objects to organize and optimize disk usage and guard against media failures. The structure is built with the objects in the following logical order:

1. Subdisks
2. Plexes (mirrors)
3. Volumes

Each object has a dependent relationship on the next-higher element, with subdisks being the lowest-level objects in the structure and volumes the highest level. LSM maintains a configuration database that describes the objects in the LSM configuration and implements utilities to manage the configuration database. Multiple mirrors, striping, and concatenation are additional techniques you can perform with the LSM objects to further enhance the capabilities of LSM.

Table 9–2 describes the LSM objects used to represent portions of the physical disks.

Table 9–2: LSM Objects

Object	Description
Volume	Represents an addressable range of disk blocks used by applications, file systems, or databases. A volume is a virtual disk device that looks to applications and file systems like a regular disk-partition device. In fact, volumes are logical devices that appear as devices in the `/dev` directory. The volumes are labeled `fsgen` or `gen` according to their usage and content type. Each volume can be composed of from one to eight plexes (two or more plexes mirror the data within the volume). Due to its virtual nature, a volume is not restricted to a particular disk or a specific area thereof. You can change the configuration of a volume (using LSM utilities) without disrupting applications or file systems using that volume.
Plex	A collection of one or more subdisks that represent specific portions of physical disks. When more than one plex is present, each plex is a replica of the volume; the data contained at any given point on each is identical (although the subdisk arrangement may differ). Plexes can have a striped or concatenated organization.
Subdisk	A logical representation of a set of contiguous disk blocks on a physical disk. Subdisks are associated with plexes to form volumes. Subdisks are the basic components of LSM volumes that form a bridge between physical disks and virtual volumes.

Table 9–2: LSM Objects (cont.)

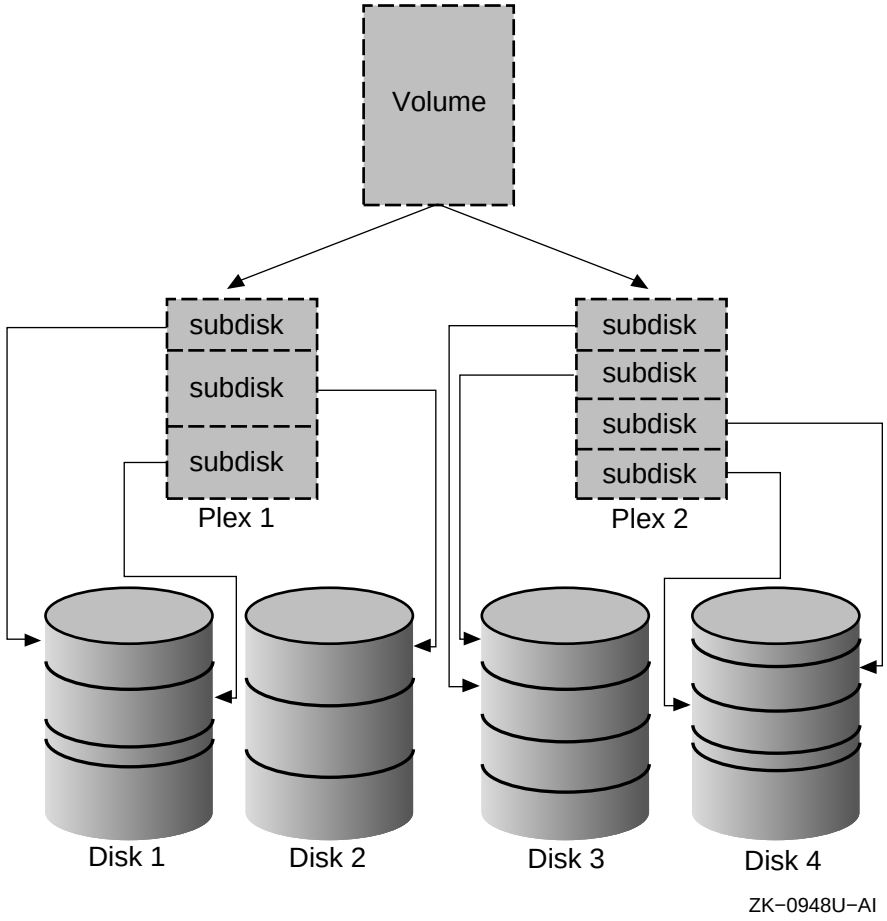
Object	Description
Disk	A collection of nonvolatile, read/write data blocks that are indexed and can be quickly and randomly accessed. LSM supports standard disk devices including SCSI and DSA disks. Each disk LSM uses is given two identifiers: a disk access name and an administrative name.
Disk Group	A collection of disks that share the same LSM configuration database. The `rootdg` disk group is a special disk group that always exists.

LSM objects have the following relationships:

- A volume consists of one to eight plexes
- A plex consists of one or more subdisks
- A subdisk represents a specific portion of a disk
- Disks are grouped into disk groups

Figure 9–2 shows an LSM configuration that includes two plexes to protect a file system or a database against data loss.

Figure 9–2: LSM Objects and Their Relationships



9.2.2 LSM Disks

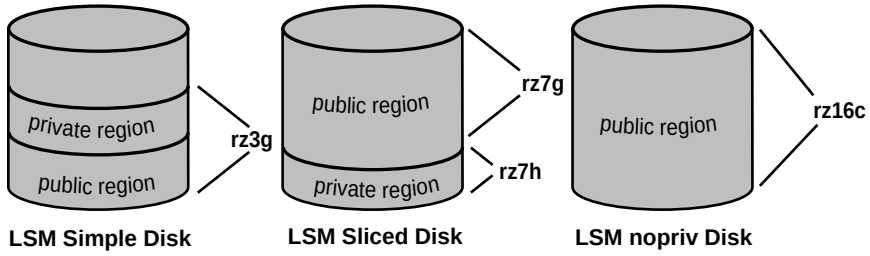
You must add physical disks to the LSM environment as LSM disks before you can use them to create LSM volumes. Refer to Section 9.6.3 and the `voldiskadd`(8) reference page for information about adding physical disks to LSM.

An LSM disk typically uses the following two regions on each physical disk:

- A small region, called the private region, in which LSM keeps its disk media label and a configuration database
- A large region, called the public region, that forms the storage space for building subdisks

Figure 9–3 illustrates the three types of LSM disks: *simple*, *sliced*, and *nopriv*. You can add all of these types of disks into an LSM disk group.

Figure 9–3: Types of LSM Disks



In Figure 9–3:

- Simple disks have both public and private regions in the same partition (`rz3g`).
- Sliced disks use the entire disk (`rz7`) and use the disk label on a disk to identify the private (`rz7h`) and the public (`rz7g`) regions.
- Nopriv disks have no private region, and so they do not contain LSM configuration information. Therefore, you can add nopriv disks only to an existing disk group that includes a simple disk or a sliced disk.

LSM configuration databases are stored on the private region of each LSM disk except the nopriv disk. The public regions of the LSM disks collectively form the storage space for application use. For purposes of availability, each simple and sliced disk contains two copies of the configuration database. A sliced disk takes up the entire physical disk, but simple and nopriv disks can reside on the same physical disk. The disk label tags identify the partitions to LSM as LSM disks.

9.2.3 Naming LSM Disks

When you perform disk operations, you should understand the disk-naming conventions for a *disk access name* and *disk media name*. Disk access names and disk media names are treated internally as two types of LSM disk objects. Some operations require that you specify the disk access name, while others require the disk media name.

The following definitions describe these disk-naming conventions:

- Disk access name (also referred to as devname or device name)

  The device name or address used to access a physical disk. A disk access name is of the form:

dd [*l*] *n* [*nnn*] [*p*]

The elements in the disk access name are described in the following table:

Element	Description
dd	A two-character device mnemonic that shows the disk type. Use ra for DSA disks and rz for SCSI disks.
[l]	The SCSI logical unit number (LUN), in the range from a to h, to correspond to LUNs 0 through 7. This argument is optional and used for SCSI Redundant Arrays of Independent Disks (RAID) devices.
n[nnn]	The disk unit number ranging from 1 to 4 digits.
[p]	The partition letter, in the range from a to h, to correspond to partitions 0 through 7. This argument is optional.

For example, rz in the device name rz3 represents a pseudonym for a SCSI disk, and rzb10h (LUN 1) represents a disk access name for a Digital SCSI RAID device having a LUN of one and using partition h.

For an LSM simple disk or an LSM nopriv disk, you must specify a partition letter (for example, rz3d). For an LSM sliced disk, you must specify a physical drive that does not have a partition letter (for example, rz3). The proper full pathname of the d partition on this simple device is /dev/rz3d. For easier reading, this document often lists only the disk access name and /dev is assumed. Also, note that you do not specify /dev in front of the device name when using LSM commands.

- Disk media name (also referred to as the disk name)

  An administrative name for the disk, such as disk01. If you do not assign a disk media name, it defaults to disk*nn*, where *nn* is a sequence number if the disk is being added to rootdg. Otherwise, the default disk media name is groupname*nn*, where groupname represents the name of the disk group to which the disk is added.

9.2.4 LSM Disk Groups

You can organize a collection of physical disks that share a common configuration or function into disk groups. LSM volumes are created within a disk group and are restricted to using disks within that disk group.

Use disk groups to simplify management and provide data availability. For example:

- On a system with a large number of disks, you might want to divide disk usage into a few disk groups based on function. This would reduce

the size of the LSM configuration database for each disk group as well as reduce the amount of overhead incurred in configuration changes.

- If a system will be unavailable for a prolonged amount of time due to a hardware failure, you can move the physical disks in a disk group to another system. This is possible because each disk group has a self-describing LSM configuration database.

All systems with LSM installed have the `rootdg` disk group. By default, operations are directed to this disk group. Most systems do not need to use more than one disk group.

Note

You do not have to add disks to disk groups when a disk is initialized; disks can be initialized and kept on standby as replacements for failed disks. Use a disk that is initialized but has not been added to a disk group to immediately replace a failing disk in any disk group.

Each disk group maintains an LSM configuration database that contains detailed records and attributes about the existing disks, volumes, plexes, and subdisks in the disk group.

9.2.5 LSM Configuration Databases

An LSM configuration database contains records describing all the objects (volumes, plexes, subdisks, disk media names, and disk access names) being used in a disk group.

Two identical copies of the LSM configuration database are located in the private region of each disk within a disk group. LSM maintains two identical copies of the configuration database in case of full or partial disk failure.

The contents of the `rootdg` configuration database is slightly different from that of an ordinary database in that the `rootdg` configuration database contains records for disks outside of the `rootdg` disk group in addition to the ordinary disk-group configuration information. Specifically, a `rootdg` configuration includes disk-access records that define the disks and disk groups on the system.

The LSM volume daemon, `vold`, uses the `volboot` file during startup to locate copies of the `rootdg` configuration database. This file may list disks that contain configuration copies in standard locations, and can also

contain direct pointers to configuration copy locations. The `volboot` file is located in `/etc/vol`.

9.2.6 Moving and Replacing LSM Disks in a Disk Group

When a disk is added to a disk group it is given a disk media name, such as `disk02`. This name relates directly to the physical disk. LSM uses this naming convention (described in Section 9.2.3) because it makes the disk independent of the manner in which the volume is mapped onto physical disks. If a physical disk is moved to a different target address or to a different controller, the name `disk02` continues to refer to it. You can replace disks by first associating a different physical disk with the name of the disk to be replaced, and then recovering any volume data that was stored on the original disk (from mirrors or backup copies).

9.3 LSM System Administration

Once a disk is under the control of LSM, all system administration tasks relating to that disk must be performed using LSM utilities and commands. For instance, if you install a file system on an LSM-controlled disk using physical disk paths rather than the LSM interfaces, LSM will be unaware that the new file system exists and will reallocate its space.

LSM provides three interfaces for managing LSM disks: a command line interface, a menu interface, and a graphical user interface. You can use any of these interfaces (or a combination of the interfaces) to change volume size, add plexes, and perform backups or other administrative tasks. You can use the LSM interfaces interchangeably. LSM objects created by one interface are fully interoperable and compatible with objects created by the other interfaces. Table 9–3 describes these LSM interfaces.

Table 9–3: LSM Administration Interfaces

Interface	Type	Description
Visual Administrator (dxlsm)	Graphical	Uses windows, icons, and menus to manage LSM volumes. The dxlsm graphical interface requires a workstation. The interface interprets the mouse-based icon operations into LSM commands. The Visual Administrator (dxlsm) interface requires the LSM software license.
Support Operations (voldiskadm)	Menu	Provides a menu of disk operations. Each entry in the main menu leads you through a particular operation by providing you with information and asking you questions. Default answers are provided for many questions. This character-cell interface does not require a workstation.
Command line	Command	Provides two approaches to LSM administration. With the top-down approach, you use the LSM `volassist` command to automatically build the underlying LSM objects. With the bottom-up approach, you use individual commands to build individual objects to customize the construction of an LSM volume.

9.4 LSM System Administration Commands

The following sections summarize some useful commands from the command line interface. Examples of how to use some of these commands are included in Section 9.6.

See also the appropriate reference pages and the manual *Logical Storage Manager* for detailed information and examples.

9.4.1 Top-Down Command

The top-down approach to managing storage means placing disks in one large pool of free storage space. You then use the `volassist` utility to specify to LSM what you need, and LSM allocates the space from this free pool. You can use `volassist` to create, mirror, grow, or shrink a volume. With `volassist`, you can use the defaults that the utility provides, or you can specify volume attributes on the command line.

The `volassist` command has the following syntax:

volassist [-b] [-g *diskgroup*] [-U *usetype*] [-d *file*] *keyword argument...*

9.4.2 Bottom-Up Commands

The bottom-up approach to storage management allows you to control the placement and definition of subdisks, plexes, and volumes. Bottom-up commands allow a great deal of precision control over how LSM creates and connects objects together. You should have a detailed knowledge of the LSM architecture before using these commands.

Bottom-up commands include `volmake` to create LSM objects, and `volume, volplex,` and `volsd` to manipulate volume, plex, and subdisk objects. The syntax for these commands is as follows:

volmake [-U *usetype*] [-o *useopt*] [-d *file*] [*type name*|[*attribute*]]...

volume [-U *usetype*] [-o *useopt*] [-Vq] *keyword argument...*

volplex [-U *usetype*] [-o *useopt*] [-V] [-v *volume*] *keyword argument...*

volsd [-U *utype*] [-o *uopt*] [-V] [-v *volume*] [-p *plex*] *keyword argument...*

9.4.3 Information Command

The `volprint` command, which has built-in parsing and formatting features, displays most of the LSM configuration and status information. The `volprint` command has the following syntax:

volprint [-AvpsdGhnlafmtqQ] [-g *diskgroup*] [-e *pattern*] [-D *database*] [-F[*type*:]*format-spec*] [*name...*]

9.5 Planning an LSM Configuration

Before setting up LSM volumes, plexes, and subdisks, you should consider the needs of your site, the hardware available to you, and the rationale for creating volumes and disk groups.

Table 9–4 presents some configuration options and describes the planning considerations that apply to LSM configurations.

Table 9–4: LSM Configuration Options

Configuration	Description
Concatenated volumes	You concatenate multiple LSM disks together to form a big volume. You can use a concatenated volume to store a large file or file systems that span more than one disk. Disk concatenation frees you from being limited by the actual physical sizes of individual disks so that you can combine the storage potential of several devices. Use the default disk group, `rootdg`, to create a concatenated volume from the public regions available. You can also add more LSM disks and create volumes from the new disks you added.
Mirrored volumes	You associate multiple plexes with the same volume to create a mirrored volume. If you are concerned about the availability of your data, then plan to mirror data on your system. You should map plexes that are associated with the same volume to different physical disks. For systems with multiple disk controllers, you should map a volume's plexes to different controllers. The `volassist` command will fail if you specify a device that is already in the volume as the mirrored plex; the bottom-up commands will not fail.
Striped volumes	For faster read/write throughput, use a volume with a striped plex. On a physical disk drive, the drive performs only one I/O operation at a time. On an LSM volume with its data striped across multiple physical disks, multiple I/Os (one for each physical disk) can be performed simultaneously. The basic components of a striped plex are the size of the plex in multiples of the stripe width used, the actual stripe width, and number of stripes. Stripe blocks of the stripe width size are interleaved among the subdisks, resulting in an even distribution of accesses among the subdisks. The stripe width defaults to 128 sectors, but you can tune the size to specific application needs. The `volassist` command automatically rounds up the volume length to multiples of stripe width.
Mirrored and striped volumes	Use mirrored and striped volumes when speed and availability are important. LSM supports mirroring of striped plexes. This configuration offers the improved I/O performance of striping while also providing data availability. The different striped plexes in a mirrored volume do not have to be symmetrical. For instance, a three-way striped plex can be mirrored with a two-way striped plex as long as the plex size is the same. Reads can be serviced by any plex in a mirrored volume. Thus, a mirrored volume provides increased read performance. However, LSM issues writes to all plexes in a mirrored volume. Because the writes are issued in parallel, there is a small amount of additional overhead as the result of a write I/O to a mirrored volume.

9.6 Implementing an LSM Configuration

After installing and licensing the LSM software (as described in the *Installation Guide*), you can use the information in the following sections to quickly get LSM up and running.

The following sections provide quick reference information to help you reenable LSM after an installation, start up LSM for the first time, and perform several common LSM operations. The examples provided use the command-line interface. See the *Logical Storage Manager* guide for complete information about using the command line interface, and for information about the LSM graphical user interface and menu interface.

9.6.1 Reenabling LSM

If you are already running LSM and the `rootdg` disk group is already initialized, you do not need to reenable LSM. For example, if you performed an upgrade installation, skip this section.

If you had LSM initialized on a system before doing a full installation, you can reenable the LSM configuration by performing the following steps:

1. Copy the `/etc/volboot` file from a backup:

   # **cp /backup/volboot /etc/volboot**

2. Create the LSM special device file:

   # **/sbin/volinstall**

3. Start the LSM daemons and volumes:

   # **/sbin/vol-startup**

9.6.2 Setting up LSM

If you are setting up LSM for the first time, you can use the `volsetup` utility to initialize LSM and create the LSM configuration database for the first time. Then, use the `voldiskadd` utility to add more disks into LSM. This is the simplest method to set up an LSM configuration.

The `volsetup` utility automatically modifies disk labels, initializes disks for LSM, creates the default disk group, `rootdg`, and configures disks into the `rootdg` disk group. You invoke the `volsetup` utility only once. To later add more disks, use the `voldiskadd` utility.

The `volsetup` utility prompts you to estimate how many disks will be managed by LSM. The utility uses the estimate to define optimal values for

the private region size (in sectors), and the number of configuration and log copies per disk.

Follow these steps to use volsetup:

1. If you are in single-user mode, set the host name for your system before initializing LSM.
2. Execute the /sbin/volsetup interactive utility by entering the following command:

   # **/sbin/volsetup rz1**

   In this example, the rz1 disk is used to initialize the rootdg disk group. If you do not give the name of a disk, LSM prompts you for one.

   Note

   When you are first setting up LSM, do not include the boot disk in the disks you specify to volsetup. After you initialize LSM, you can encapsulate the root and swap partitions and add them to the rootdg disk group or another disk group.

3. The volsetup utility modifies the /etc/inittab file. When the system reboots, LSM is started automatically by the initialization process when it reads the LSM entries in the inittab file. (See inittab(4) for more information.)
4. The LSM /sbin/lsmbstartup script starts the LSM vold daemon and the voliod error demon. After running the volsetup procedure, check that the vold daemon is running.

The volsetup utility creates the /etc/vol/volboot file. This file is used to locate copies of the rootdg disk group configuration when the system starts up.

Note

Do not delete or manually update the /etc/vol/volboot file; it is critical for starting LSM.

9.6.3 Adding a Disk to a Disk Group

Once LSM has been initialized with the /sbin/volsetup utility, you can add more physical disks or disk partitions to the rootdg disk group or add

a new disk group by executing the interactive `voldiskadd` utility. This utility requires that a disklabel already exist on the device. Refer to the `disklabel`(8) reference page for complete information. For example, you could add a disk partition to the `rootdg` disk group by executing the following command:

```
# voldiskadd rz3
```

To initialize a disk without adding it to a disk group, use the `voldisksetup`(8) command. This command allows you to add an LSM simple disk or sliced disk.

To add a physical disk to LSM with a specific private region size, use the `voldisksetup`(8) command. For example, use the following command to initialize a sliced LSM disk with a private region size of 2048 sectors:

```
# voldisksetup -i rz3 privlen=2048
```

Use the `voldg` command to add the LSM disk to a disk group.

9.6.4 Creating a Volume in a Disk Group

After you create a disk group and add disks, use the `volassist` command to create volumes. For example:

```
# volassist -g disk_group make volume length attribute=value
```

To create a volume in a disk group, use the instructions in the following list, or use the dxlsm graphical user interface (GUI).

- To use nonreserved disks to create a 10 MB volume in the `rootdg` disk group, enter the following command:

  ```
  # volassist -g rootdg make vol01 10m
  ```

- To use nonreserved disks to create a 1024 Kb volume in the `dg1` disk group, enter the following command:

  ```
  # volassist -g dg1 make vol02 1024k
  ```

- To create a volume on a specified disk in the `rootdg` disk group, enter the following command:

  ```
  # volassist -g rootdg make vol03 200000s rz7
  ```

- To use nonreserved disks to create a 200,000 sector volume in the `rootdg` disk group and exclude the rz9 disk, enter the following command:

  ```
  # volassist -g rootdg make vol03 200000s !rz9
  ```

- To create a 20 MB striped volume from the `rootdg` disk group using three LSM disks with a stripe width of 64 Kb (the default), enter the following command:

```
# volassist -g rootdg make vol04 20m layout=stripe nstripe=3
```

9.6.5 Mirroring a Volume

Once a volume is created and enabled, use the `volassist` utility to create and attach new plexes to the volume.

- The following command creates three plexes of the vol02 volume in the dg1 disk group. The command is executed in the background because it may take a long time for the command to complete:

  ```
  # volassist -g dg1 mirror vol02 nmirror=3 &
  ```

- The following command creates a 30 MB mirrored volume named vol05 from the `rootdg` disk group. The `mirror=yes` option specifies the number of mirrors as two. This is the default.

  ```
  # volassist -g rootdg make vol05 30m mirror=yes
  ```

9.6.6 Changing the Size of a Volume

You can use the `volassist` utility to increase or decrease the size of a volume. To change the size of a volume, use the following examples as guidelines:

- Enter the following command to increase the size of the vol01 volume by 2 MB:

  ```
  # volassist growby vol01 2m
  ```

- Enter the following command to decrease the size of the vol01 volume by 1024 Kb:

  ```
  # volassist shrinkby vol01 1024k
  ```

Caution

The following restrictions apply to grown LSM volumes:

- A volume containing one or more striped plexes cannot grow in size.
- Neither UFS nor AdvFS file systems can take advantage of the extra space in a grown LSM volume.
- Shrinking an LSM volume with either a UFS or AdvFS file system causes loss of data.

10

Administering User Accounts and Groups

Adding, modifying, and removing individual user accounts and groups of users is a routine but important activity that a system administrator frequently performs.

After introducing user account and group administration, this chapter describes the following tasks:

- Adding a user account
- Changing information in a user account
- Removing a user account
- Adding and removing a group

Note

You can also use the *SysMan* `dxaccounts` command to perform these tasks.

10.1 Understanding User Accounts and Groups

Administering user accounts and groups involves managing the contents of the system's password and group files. On standalone systems, the files you manage are `/etc/passwd`, which is documented in `passwd`(1), and `/etc/group`, which is documented in `group`(4).

On networked systems, typically, the Network Information Service (NIS) is for central account and group management. NIS allows participating systems to share a common set of password and group files. See the *Network Administration* manual for more information.

If enhanced security is enabled on your system, you need to administer more than the `/etc/passwd` file for security. For example, the protected password database is used for security related information such as minimum password lengths and password expiration times. These tasks are documented in the *Security* manual.

10.1.1 The Password File

The passwd file for a standalone system identifies each user (including root) on your system. Each passwd file entry is a single line that contains seven fields. The fields are separated by colons and the last field ends with a new-line character. The syntax of each entry and the meaning of each field is as follows:

username: *password*: *user_id*: *group_id*: *user_info*: *login_directory*: *login_shell*

username — The name for the user account. The *username* must be unique and consist of from one to eight alphanumeric characters.

password — You cannot enter a password directly. Enter an asterisk (*) in the passwd field to disable a login to that account. An empty password field allows anyone who knows the login name to log in to your system as that user. Refer to Section 10.2.2.4 for instructions on assigning a user password with the passwd command.

user_id — The UID for this account. This is an integer between 0 and 32767 and must be unique for each user on the system. Reserve the UID 0 for root. Assign each UID in ascending order beginning with 100. Lower numbers are used for pseudousers such as bin or daemon.

group_id — The GID for this account. This is an integer between 0 and 32767. Reserve the GID 0 for the system group. Be sure to define the GID in the group file.

user_info — This field contains additional user information such as the full user name, office address, telephone extension, and home phone. The finger command reads the information in the *user_info* field. Users can change the contents of their *user_info* field with the chfn command. Refer to Section 10.3.2, as well as the finger(1) and chfn(1) reference pages for more information.

login_directory The absolute pathname of the directory where the user account is located immediately after login. The `login` program assigns this pathname to the HOME environment variable. Users can change the value of the HOME variable, but if a user changes the value, then the home directory and the login directory are two different directories. Create the login directory after adding a user account to the `passwd` file. Typically the user's name is used as the name of the login directory. Refer to the `chown`(1), `mkdir`(1), `chmod`(1), and `chgrp`(1) reference pages for additional information on creating a login directory.

login_shell The absolute pathname of the program that starts after the user logs in. Normally, a shell starts. If you leave this field empty, the Bourne shell `/bin/sh` starts. Refer to the `sh`(1) reference page for information on the Bourne shell. Users can change their login shell by using the `chsh` command. Refer to Section 10.3.3 and the `chsh`(1) reference page for more information.

10.1.2 The Group File

All users are members of at least one group. The `group` file identifies the group name for a user. There are two primary reasons to group user accounts:

- Several users work together on the same files and directories; grouping these users together simplifies file and directory access.
- Only certain users are permitted access to system files or directories; grouping them together simplifies the identification of privileged users.

The `group` file is used for the following purposes:

- To assign a name to a group identification number used in the `passwd` file
- To allow users to be members of more than one group by adding the user account to the corresponding group entries

Each entry in the `group` file is a single line that contains four fields. The fields are separated by colons, and the last field ends with a new-line character. The syntax of each entry and the meaning of each field is as follows:

groupname: *password*: *group_id*: *user1* [*user2*,..., *userN*]

groupname	The name of the group defined by this entry. The *groupname* consists of from one to eight alphanumeric characters and must be unique.
password	Place an asterisk (*) in this field. Entries for this field are currently ignored.
group_id	The group identification number (GID) for this group. This is an integer between 0 and 32767. Reserve the GID 0 for the system. The GID must be unique.
user	The user account belonging to this group as defined in the `passwd` file. If more than one user belongs to the group, the user accounts are separated by commas. The last user account ends with a new-line character. A user can be a member of more than one group.

There is a limitation on the number of groups that a user can be in, as documented in `group`(4). The maximum line length is LINE_MAX as defined in the `limits.h` file. Digital recommends that user accounts be divided into a number of manageable groups.

10.1.3 The Administrative Tools

There are several tools you use to administer user accounts and groups:

- The Account Manager, which provides a graphical user interface for managing system envrironments with standard security or enhanced security features. The Account Manager can also manage the NIS account databases. For more information on using the Account Manager, please refer to the application's online help and the `dxaccounts`(1X) reference page.

  Note

  The Account Manager is the preferred graphical interface for managing user accounts and groups. It replaces the `Xsysadmin` and `XIsso` commands.

- Several commands provide a command line interface to user account and group management, including user accounts on systems using NIS

and enhanced security. The commands are documented in the following reference pages: useradd(8), usermod(8), userdel(8), groupadd(8), groupmod(8), and groupdel(8).

- The adduser and addgroup utilities, documented in adduser(8) and addgroup(8). These utilities provide simple, interactive scripts you can use to add new user accounts and groups. These utilities can be used only on systems that do not use NIS.
- The vipw utility, documented in vipw(8), allows you to invoke an editor in order to edit the password file manually. You can use the utility to edit the local password database, but you cannot use it to edit the NIS database. Additionally, you cannot use the vipw utility on systems that have enhanced security. The vipw command allows you to edit the passwd file and at the same time locks the file to prevent others from modifying it. This command also does consistency checks on the password entry for root and does not allow a corrupted root password to be entered into the passwd file.

10.2 Adding a User Account

This section describes how to:

- Add a user account with the adduser utility
- Add a user account manually

10.2.1 Adding a User Account with the adduser Utility

The adduser utility automates the process of adding a user account. This utility performs the following tasks:

1. Adds a user account to the system password file.
2. Sets the user's primary and secondary groups.
3. Creates the user's home directory and copies the contents of /etc/skel to this new directory. The /etc/skel file contains a default set of basic files for a new user. Specifically, it contains the following files:
   - For C shell users, there are sample .login and .cshrc files
   - For Bourne and Korn shell users, there is a sample .profile file
   - A bin directory file
4. Checks to see if the /var/spool/mail directory exists. If the directory does not exist, the adduser utility creates it.

When you invoke the adduser utility, it responds with a series of messages and prompts you for the following information:

- Login name. If the login name already exists, the utility exits.
- User identification number (UID). The utility displays a default UID that is based on the existing UIDs in the `/etc/passwd` file.
- Full name of the user.
- Login group. If the login group does not exist, the utility displays a list of the existing groups and asks if you want to add a group to the `group` file. Refer to Section 10.2.2.5 for information on adding a group.
- Parent directory.
- Login shell.
- Password.
- Whether to use the hashed password database.

To use the `adduser` utility:

1. Log in as root and enter the following command at the prompt:

   # **adduser**

   The utility responds with a series of prompts and messages. The brackets ([ ])indicate a default response. Press Return to accept the default or enter a different response and press Return, as shown in the following example:

```
Enter a login name for the new user (for example, john): chris
Enter a UID for (chris) [5006]:   Return
Enter a full name for (chris): Christopher Ryan
Enter a login group for (chris) [users]:   Return
Enter another group that (chris) should be a member of.
(<Return> only if none):   Return
Enter a parent directory for (chris) [/usr/users]:    Return
The shells are:
/usr/bin/sh       /usr/bin/ksh      /bin/csh          /bin/ksh
/usr/bin/csh      /bin/sh
Enter a login shell for (chris) [/bin/sh]:   Return
```

2. If your system is running enhanced security, the `adduser` utility asks if you want to edit the protected password entry for the user. If the EDITOR environment variable is set, the `adduser` utility uses $EDITOR to edit the user's protected `passwd` entry. Refer to `prpasswd`(4) and `authcap`(4) for more information about the fields in the protected password database.
3. The `adduser` utility displays informational messages and prompts you for a new password for the user. To ensure confidentiality, the password is not displayed.

```
Adding new user ...
Rebuilding the password database...
10 password entries, maximum length 145
Creating home directory...
```

```
You must enter a new password for (chris).
Changing password for chris.

New password:
Retype new password:
```

If you mistype the password during verification, no password is set and the account is disabled. To enable the user account, enter the passwd command followed by the user name.

4. If a hashed passwd database did not exist previously, the program prompts you to create one:

```
The hashed password database does not exist.
Do you want to create it ([y]/n)?
```

To create a hashed passwd database, enter yes at the prompt. The adduser utility creates one for you. If you do not want a hashed passwd database, enter no at the prompt. Refer to vipw(8) for information about editing the /etc/passwd file.

Note

A hashed passwd database allows for faster lookups of password file data and thus can improve system performance. Digital recommends you use a hashed passwd database.

10.2.2 Adding a User Account Manually

To add a user account manually:

1. Add an entry for the user to the passwd file by using the vipw command.
2. Add an entry for the user account to the group file.
3. Supply the default shell scripts for the user's working environment.
4. Assign a password to protect the user account.
5. Verify the accuracy of the group and passwd files.

The following sections describe these tasks in detail.

10.2.2.1 Adding a User Account to the passwd File

Note

You cannot use the vipw utility to edit the protected password database on systems running with enhanced security. For these systems, you should use the adduser utility, the useradd

command, or the Account Manager graphical interface to edit the passwd file.

To edit the passwd file:

1. Log in as root.
2. Enter the vipw command to add the required line entry to the passwd file:

```
# vipw
root:TZVtfX5VbS3KY:0:1:System PRIVILEGED Account,,,:/:/bin/sh
daemon:*:1:daemon
uucp:*:2:uucp
.
.
marcy:*:201:20:Marcy Swanson,dev,x1234:/usr/users/marcy:/bin/sh
```

   The previous example shows that user marcy has a UID of 201 and a GID of 20. The login directory is /usr/users/marcy and the Bourne shell (/bin/sh) is defined as the login shell. Since the password field contains an asterisk (*), user marcy cannot log in to the system. Section 10.2.2.4 describes how to add a password to the passwd file.

3. Close the file.

If a hashed passwd database exists, vipw uses the mkpasswd command to re-create it. A hashed passwd database is an indexed database that allows for faster searches of the passwd file. The following example shows the message displayed after closing the passwd file where a hashed passwd database existed previously:

```
10 password entries, maximum length 88
```

If a hashed passwd database does not exist, a message is displayed informing you that passwd it does not exist and asks if you want a database created. If you want a hashed passwd database, enter yes at the prompt. If you do not want a hashed passwd database, enter no at the prompt. Refer to vipw(8) for more information.

Note

In an NIS environment you can add a user account to either the local passwd file or the NIS distributed passwd file. Accounts added to the local passwd file are visible only to the system to which they are added. Accounts added to the NIS distributed passwd file are visible to all NIS clients that have access to the distributed file. Refer to nis_manual_setup(7) for more information on adding users in a distributed environment.

10.2.2.2 Adding an Entry to the group File

To add a new group or a user to an existing group, add a line entry to the `group` file, as follows:

1. Log in as root and change to the `/etc` directory.
2. Use the `cp` command to copy the `group` file to a temporary file. For example, enter:

   ```
   # cp group group.sav
   ```

3. Open the `group` file and add the required line entry. Be sure to include all four fields in this entry. A file is displayed similar to the following, which shows that users `diaz`, `kalle`, `marcy`, and `chris` belong to the `users` group that has a GID of 15:

   ```
   system:*:0:root,diaz,kalle,marcy
   daemon:*:1:daemon
   uucp:*:2:uucp
   .
   .
   .
   users:*:15:diaz,kalle,marcy,chris
   ```

4. Close the file.
5. Use the `vipw` command to edit the `passwd` file to include the GID in the *group_id* field of each user who is a member of the group. Refer to Section 10.2.2.1 for more information about the `passwd` file.

If at a later date you change the group a user belongs to, be sure to change the parent directory's GID also.

10.2.2.3 Providing the Default Shell Scripts

Users can customize their working environment by modifying their startup files. When a user logs in to the system, the login shell looks for startup files in the login directory. If the shell finds a startup file, it reads the file and executes the commands.

Table 10–1 displays each shell and the corresponding startup files.

Table 10–1: Shells and Their Startup Files

Shell	System Startup File	Login Startup Files
`/bin/csh`	`/etc/csh.login`	`.cshrc`, `.login`
`/bin/ksh`	`/etc/profile`	`.profile`
`/bin/sh`	`/etc/profile`	`.profile`

The operating system uses these startup files to initialize local and global environment variables, shell variables, and terminal types. Use the following procedure to copy the startup files to the login directory of each user account:

1. Copy the startup files for each shell to the user's login directory by using the cp command. For example, to copy the startup files to the user marcy directory, enter:

   ```
   # cd /usr/skel
   # cp -R 'ls -A' /usr/users/marcy
   ```

2. Change to the user's login directory and change file ownership and access permissions from root to the user for each file. For example, to make these changes to all of the files beginning with dot (.), for user marcy, enter the following sequence of commands:

   ```
   # cd /usr/users/marcy
   # chmod 755 .??*
   # chown marcy .??*
   ```

3. To confirm that the changes were made, use the ls command to list marcy's files:

   ```
   # ls -Al
   ```

Refer to the csh(1), ksh(1), and the sh(1) reference pages for more information on the shell commands.

10.2.2.4 Assigning a Password

Use the passwd command to assign a password for a user account. When you enter the passwd command, the program prompts you for a password. Each password must have at least five characters, but not more than eight, and can include digits, symbols, and the characters of your alphabet. The password cannot be all lowercase characters. The passwd command encrypts the specified password and inserts it in the password field of the passwd file.

To assign an initial password, use the following syntax:

passwd *username*

For example, to assign an initial password for user marcy, enter the following command:

```
# passwd marcy
```

The system responds with the following prompts. Enter and verify the new password for the user. To ensure confidentiality, the password will not be displayed.

```
Changing password for marcy.

New password:
Please don't use an all-lower case password.
Unusual capitalization, control characters or digits are suggested.
New password:
Retype new password:
```

If a hashed passwd database is not in use, the system displays the following informational message:

```
Hashed database not in use, only /etc/passwd text file updated.
```

A hashed passwd database is an indexed database that allows for a faster search of the passwd file.

10.2.2.5 Verifying the Accuracy of the group and passwd Files

Once you have completed all the tasks for adding a user account, use the grpck and the pwck commands to check the accuracy of the group and passwd files.

Note

If your system is running enhanced security, you should also use the authchk utility to verify the accuracy of the protected password database.

The grpck command verifies that the number of fields, group name, GID, and all login names that appear in the passwd file are correct. If any fields are incorrect, grpck writes the inconsistencies to standard output. For example:

```
# grpck
users:*:15:diaz,kalle,marcy,chris,farkle
        farkle - Logname not found in password file     [1]
mem:*:3:
        Null login name     [2]
+:
        Too many/few fields     [3]
```

[1] Refer to Section 10.2.2.1 for information on adding a user account to the passwd file.

[2] Ignore this message.

[3] Ignore this message. These characters are necessary for running NIS.

Refer to the grpck(8) reference page for more information.

The pwck command checks for any inconsistencies in the passwd file. The pwck command verifies the number of fields, login name, UID, GID,

existence of a login directory, and optional program name. If any of the fields are missing, pwck writes the inconsistencies to standard output. For example:

```
# pwck
nobody:*Nologin:4294967294:4294967294:anonymous NFS user:/:
        Invalid UID     1
        Invalid GID     2
        Optional shell file not found     3
```

1 Refer to Section 10.2.2.1 for valid UID numbers.

2 Refer to Section 10.2.2.1 for valid GID numbers.

3 Ignore this message.

Refer to the pwck(8) reference page for more information.

10.3 Changing Information in a User Account

This section describes how to change information about a user account. The following tasks are discussed:

- Changing passwords
- Changing the user_info field
- Changing the login shell
- Setting disk quotas

10.3.1 Changing Passwords

You should periodically change the root password. This protects the system from access by system users who should not have root access, as well as from external intruders.

There may be times when a user forgets his or her password. If this happens, change the user's password as described in Section 10.2.2.4 and tell the user the new password.

10.3.2 Changing the user_info Field

The user_info field in the passwd file contains the name, room number, office phone, and home phone of the user. To change this information, use the chfn command with the following syntax:

chfn [*username*]

For example, to change the information for user marcy, enter:

```
% chfn marcy
```

The system displays information similar to the following example. The brackets ([ ])indicate a default response. Press Return to accept the defaults or enter a different response and press Return.

```
Default values are printed inside of '[]'.
To accept the default, type <return>.
To have a blank entry, type the word 'none'.

Name [Marcy Swanson]: Return
Room number (Exs: 597E or 197C) []: Return
Office Phone (Ex: 6426000) []: 3311
Home Phone (Ex: 9875432) []: Return
```

10.3.3 Changing the Login Shell

There may be a time when you want to change a user's login shell. To see a list of the shells the user is allowed to select from, enter the following command:

```
# cat /etc/shells
```

The system prints a list similar to the following:

```
/bin/sh
/bin/csh
/bin/ksh
```

To change a user's login shell, use the `chsh` command with the following syntax:

chsh [*username*]

For example, to change user `marcy's` login shell from the Bourne shell to the C shell, enter:

```
# chsh marcy
```

The system responds with the following information. At the prompt, enter the new shell user `marcy` will be using. For example:

```
Old shell: /bin/sh
New shell: /bin/csh
```

The next time user `marcy` logs in, she will be using the `/bin/csh` shell.

10.3.4 Setting File System Quotas

If you configured your system with file system quotas (also called disk quotas), you can set a quota for the number of inodes or disk blocks allowed for each user account or group on your system. To optimize disk space and to save yourself some work, set quotas by grouping user accounts according

to their need for disk space. The following information is specific to the UNIX File System (UFS). If you are using the POLYCENTER Advanced File System (AdvFS), refer to Chapter 8.

10.3.4.1 Understanding User Account and Group Quota Limits

You set quotas for user accounts and groups by file system. For example, a user account can be a member of several groups on a file system and also a member of other groups on other file systems. The file system quota for a user account's files on that file system. A user account's quota is exceeded when the number of blocks (or inodes) used on that file system are exceeded.

Like user account quotas, a group's quota is exceeded when the number of blocks (or inodes) used on a particular file system is exceeded. However, the group blocks or inodes used only count toward a group's quota when the files that are produced are assigned the GID for the group. Files that are written by the members of the group that are not assigned the GID of the group do not count toward the group quota.

Note

Quota commands display block sizes of 1024-byte blocks.

10.3.4.2 Setting File System Quotas for User Accounts

To set a disk quota for a user, you can create a quota prototype or you can use an existing quota prototype and replicate it for the user. A quota prototype is an equivalence of an existing user's quotas to a prototype file, which is then used to generate identical user quotas for other users. Use the `edquota` command to create prototypes. If you do not have a quota prototype, create one by following these steps:

1. Log in as root and use the `edquota` command with the following syntax:

   edquota *proto-user users*

   For example, to set up a quota prototype named `large` for user `eddie`, enter the following command:

   ```
   # edquota large eddie
   ```

   The program creates the `large` quota prototype for user `eddie`. You must use a real login name for the *users* argument.

2. Edit the quota file opened by the `edquota` program to set quotas for each file system that user `eddie` can access.

To use an existing quota prototype for a user:

1. Enter the edquota command with the following syntax:

   edquota –**p** *proto-user users*

   For example, to set a disk quota for marcy, using the large prototype, enter:

   # **edquota -p large marcy**

2. Confirm that the quotas are what you want to set for user marcy. If not, edit the quota file and set new quotas for each file system that user marcy can access.

Refer to quota(1) and edquota(8) for more information.

10.4 Removing a User Account

To remove a user's account, you must remove all the files and directories from the account and rename the user's entry for the group and passwd files. You can rename an account manually or by using the removeuser utility.

10.4.1 Removing a User Account with the removeuser Utility

The removeuser utility automates the process of removing a user account. This utility performs the following tasks:

1. Removes the user's entry from the /etc/passwd file and any references to the user's account from the /etc/group file
2. Searches several administrative directories and files for occurrences of the user and informs you if they exist
3. Allows the removal of the home directory, which includes directories and files, and mail files

To use the removeuser utility, log in as root. At the prompt, enter:

removeuser

The program responds with a series of prompts and messages, as shown in the following example:

```
Enter a login name to be removed or <RETURN> to exit: kalle
This is the entry for (kalle) in the /etc/passwd file:

 kalle:/v7ZY9/tF1z5w:12:15:Kalle Anderson:/usr/users/kalle:/ksh
Is this the entry you want to delete (y/n)? y
Working ...
Entry for (kalle) removed.
```

```
Searching relevant directories and files for user (kalle) ...
None found.
Do you want to remove the home directory, all subdirectories,
files and mail for (kalle) (y/n)? y
The files for (kalle) will be lost if not backed up.
Are you sure you want to remove these files (y/n)? y
Removing /usr/users/kalle

Removing /usr/spool/mail/kalle

Finished removing user account for (kalle)
```

10.4.2 Removing a User Account Manually

To manually remove a user account from your system:

1. Remove the user's files and directories.
2. Remove the user's entry from the `group` file.
3. Remove the user's entry from the `passwd` file.
4. Remove the user's `/usr/spool/mail/username` file.

The following sections describe each task and provide instructions for removing the files and directories.

10.4.3 Removing a User's Files and Directories

Before removing files or directories from the user's account, follow these steps:

1. Make sure that the associated files and directories are not being used by other users on your system.
2. Back up the user's login directory to diskette or tape. Refer to Chapter 12 for more information.

To remove a user's files and directories:

1. Use the `rm -r` *`login_dir`* command to remove the user's login directory (including all of the directory's files and subdirectories). For example, to remove the login directory (including all of the files and subdirectories) for user `marcy`, enter:

   # **`rm -r /usr/users/marcy`**

2. Use the `rm` *`mail_dir`* command to remove the user's mail directory. For example, to remove the mail file for user `marcy`, enter:

   # **`rm /usr/spool/mail/marcy`**

3. Use the find command to ensure that no files remain that were owned by the user. For example, to verify that user marcy no longer owns files, enter:

   # **find /usr/users -user marcy -print**

   The find command locates user files that are links (identified by a notation of >1), user files within directories (identified by a notation of 1), or user directories (identified by a notation of 2). Refer to find(1) for more information.

4. If the find command locates any user files or directories, use the chown command to change the ownership to a different user (one who still needs to access the file). If there is no reason to save or maintain these files, remove them.

5. Remove the user's crontab and atjobs files if they exist. For example:

   # **rm /var/spool/cron/crontabs/marcy**
   # **rm /var/spool/cron/atjobs/marcy**

10.4.4 Removing a User's Account from the group File

Since users can be members of more than one group, modify all line entries in the group file that contain the user name within the *user* field.

To modify a group file entry:

1. Log in as root and change to the /etc directory.

2. Use the cp command to copy the group file to a temporary file.

   # **cp group group.sav**

3. Open the group file and remove the user's name from each line entry in which it is listed. The screen displays a file similar to the following, which shows that user marcy is not a member of the users group:

```
system:*:0:root,diaz
daemon:*:1:daemon
uucp:*:2:uucp
⋮
users:*:15:diaz,chris
⋮
```

4. Close the file.

10.4.5 Removing a User's Account from the passwd File

After you remove a user's account from the passwd file, the system can no longer identify the user. When removing an account for a user, use the

vipw command to delete the line entry that identifies the user. The vipw command allows you to edit the passwd file and at the same time locks the file to prevent others from using it. Refer to Section 10.2.2.1 for information on editing the passwd file.

If you maintain accounting on a monthly basis, do not remove the line entry for the user's account from the passwd file until the monthly accounting has been done. Since the accounting commands access the passwd file, removing the user account line entry will create inaccuracies in your accounting.

However, since your primary goal is to restrict the user from gaining access to the system, you can immediately suspend the user from logging in by substituting NO_LOGIN for the encrypted user password in the passwd file. For example, the line entry for user marcy is as follows:

```
marcy:IK7Nv8f86Jo:201:20:Marcy Swanson,dev,x1234:/usr/users/marcy:/bin/csh
```

Replace the encrypted password with NO_LOGIN as shown in the following example:

```
marcy:NO_LOGIN:201:20:Marcy Swanson,dev,x1234:/usr/users/marcy:/bin/csh
```

To disable network logins, delete the user's account from any proxy files such as the user's .rhosts file.

10.5 Adding and Removing Groups

This section describes how to:

- Add a group with the addgroup utility
- Add a group manually
- Remove a group

10.5.1 Adding a Group with the addgroup Utility

The addgroup utility automates the process of adding a group to the /etc/group file.

When you invoke the addgroup utility, the program responds with a series of prompts and messages asking you for the following information:

- Group name
- Group identification number (GID)

To use the addgroup utility, log in as root and enter the following command at the prompt:

```
# addgroup
```

The program responds with a series of prompts and messages. The brackets ([]) indicate the default response. Press Return to accept the default or enter a different response and press Return, as shown in the following example:

```
Enter a new group name or <Return> to exit: newgroup
Enter a new group number [112]:  Return
Group newgroup was added to the /etc/group file.
```

The addgroup utility adds the new group to the /etc/group file.

10.5.2 Adding a Group Manually

To add a new group, add a line entry to the group file:

1. Log in as root and change to the /etc directory.
2. Use the cp command to copy the group file to a temporary file. For example, enter:

   ```
   # cp group group.sav
   ```

3. Open the group file and add the required line entry. Be sure to include all four fields in this entry. A file is displayed similar to the following, which shows that users diaz, kalle, marcy, and chris belong to the users group that has a GID of 15:

   ```
   system:*:0:root,diaz,kalle,marcy
   daemon:*:1:daemon
   uucp:*:2:uucp
   ⋮
   users:*:15:diaz,kalle,marcy,chris
   ```

4. Close the file.
5. Use the vipw command to edit the passwd file to include the GID in the *group_id* field of each user who is a member of the group. Refer to Section 10.2.2.1 for more information about the passwd file.

If at a later date you change the group a user belongs to, be sure to change the parent directory's GID also.

10.5.3 Removing a Group

To remove a group that no longer has any members, delete the corresponding line from the group file as follows:

1. Log in as root and edit the passwd file line entry for each member of the group by using the vipw command. You can either assign a

different group number or delete the current group number. If you assign a different group number, make sure that it corresponds to a current (or new) group entry in the group file. Refer to Section 10.2.2.1 for information on editing the passwd file.

2. Remove the original group line entry from the group file. To delete a group file entry:

   a. Log in as root and move to the /etc directory.

   b. Use the cp command to copy the group file to a temporary file.

      ```
      # cp group group.sav
      ```

   c. Open the group file and delete the appropriate group line entry.

   d. Close the file.

11

Administering the Print Services

This chapter describes how you set up and administer the files and programs that make up the Digital UNIX print services. You can set up and administer the print services immediately after a new installation or upgrade to a new version of the operating system, or you can wait until later. For example, you can wait until you have installed a printer and have gathered the information about its characteristics that you need to set it up.

The first part of this chapter describes how to use the `lprsetup` utility to add a print device and automatically set up the print environment. The second part of this chapter discusses the routine print services administration that you can perform, using either the `lprsetup` utility or by editing system files. The last part of this chapter contains detailed reference information about the `lpd` print daemon and the system files associated with the print services.

Note

You can also use the *SysMan* `printconfig` command to perform some of these tasks.

11.1 Administrative Tasks

To set up the print system in a Digital UNIX operating environment, you perform tasks such as:

- Physically connecting a printer to the system or ensuring that you have access to it through a network
- Adding information about a printer in the `/etc/printcap` file
- Creating the required device files and spooling directories
- Starting the `lpd` daemon
- Verifying printer installation and testing printing

11.2 Interfaces to Print Services

There are several ways you can administer print services:

- If you have a CDE Desktop graphics interface, you can use the Print Configuration application to administer print services. This interface is the recommended interface to the print services.
- If you do not have a CDE Desktop graphics interface, or if you wish to continue to use current methods of print service administration, you can use the `lprsetup` utility to administer print services. The `lprsetup` utility will be retired in a future release of the operating system.
- You can perform these tasks manually by creating and modifying the required files with a text editor.

11.3 Print Services Commands

Unless you are using the CDE Desktop Print Configuration application, you use the following commands to manage the print system:

- The `lprsetup` command to add, modify, and remove printers
- The `lpc` command to monitor and control printer operations
- The `lpr` command to send files to the printer
- The `lprm` command to remove print jobs from the queue
- The `lpq` command to check a print queue
- The `lpstat` command to check a print queue (similar to `lpq`)

Refer to the `lprsetup`(8), `lpc`(1), `lpr`(1), `lprm`(1), `lpq`(1), and `lpstat`(1) reference pages for more information about these commands.

After a printer is set up and running on your system, you need to:

- Manage the system and take care of routine changes such as adding new printers or changing the characteristics of existing printers
- Administer the print queues and files as your system needs change
- Control the daily operations and throughput of print jobs

11.4 Using lprsetup to Set Up the Print System

This section describes the information you need in order to use the `lprsetup` utility to connect a printer to your computer. Before proceeding, verify that the printer is physically connected to your system, accessible on the network (for remote printing), and has been tested as described in the owner's manual.

To use the `lprsetup` utility, you must have the Printer Support Environment subset installed. To see if you have this subset installed, enter:

```
# setld -i | grep OSFPRINT
```

If the OSFPRINT subset is installed, the following information is displayed:

```
OSFPRINT400     installed       Local Printer Support (Printing Environment)
```

If the OSFPRINT subset is not installed, see the *Installation Guide* for information on adding this, or any, subset with the `setld` utility.

11.4.1 Gathering Information

Before adding a printer, you need to gather the information about the printer that you will need to interact with the `lprsetup` program. The `lprsetup` program updates the information in the `/etc/printcap` file using the information you supply.

If your system is part of a network, you may need to consult your local network administrator about the correct procedure for adding a printer.

The following is a list of the information you need:

- Name of the printer (print queue)
- Printer type
- Printer device name
- Printer synonyms (alternative names)
- Printer accounting
- Spooler directory
- Error log file
- Connection type (LAT device)
- Baud rate (hard-wired ports only)

If you are adding a remote printer, you need the name of the machine the printer is connected to (host name) and the remote printer queue name.

The following sections describe how you obtain the required information.

11.4.1.1 Printer Name

The printer name is the name by which you want to identify the printer through the `lpr` command. For example:

```
# lpr -Pprintername
```

The `lprsetup` program uses an internal numbering scheme from 0 to 99. The next available number is the default name. You can choose the default by pressing the Return key or by entering any other alphanumeric name that is appropriate. The `lprsetup` program always assigns at least two

printer synonyms: the default number and 'lp default number', plus any others you specify. If the default number were 1, the two names would be '1' and 'lp1'. This printer could then be identified with 'lpr -P1' or 'lpr -Plp1'.

If you have only one printer or are entering the first of many printer names, the first name will have a printer number of 0. This is recognized as your system's default printer and will have an additional name of 'lp'. This means if you use the `lpr` command without specifying a specific printer this is the one it will use.

If this is the first printer connected to your system or a new printer added to an existing print system, create names that do not conflict with existing printer names. Ask your network administrator for the names of the remote printers on the network.

11.4.1.2 Printer Type

The printer type corresponds to the product name of the printer, such as the LN03 laser printer. If you are using the `lprsetup` program, printers are listed by type and only those supported by Digital are listed. These printers have some default values already included in the setup program.

The supported printer types are defined in Table 11–1.

Table 11–1: Supported Printer Types

Printer Name	Abbreviation
LA50	la50
LA70 Personal Printer	la70
LA75 Plus Companion Printer	la75
LA324 MultiPrinter	la324
LA424 MultiPrinter	la424
ColorMate PS	lf01r
LG02 Line Printer	lg02
DEClaser 2200	lg06
LG12 Impact Line Matrix Printer	lg12
LG31 Line Printer	lg31
LJ250 Companion Color Printer	lj250
LN03 Laser Printer	ln03
LN03 Laser Printer	ln03s
DEClaser 2100	ln05

Table 11–1: Supported Printer Types (cont.)

Printer Name	Abbreviation
DEClaser 2200	ln06
DEClaser 1100	ln07
DEClaser 3200	ln08
DEClaser 5100	ln09
LN03 PostScript Printer	ln03r
DEClaser 2150	ln05r
DEClaser 2250	ln06r
DEClaser 1150	ln07r
DEClaser 3250	ln08r
DEClaser 5100	ln08r
IBM Proprinter	ibmpro
NEC Silentwriter Model 290	nec290
Epson FX-80	fx80
Epson FX-1050	fx1050
HP LaserJet Model IIP	hpIIP
HP LaserJet Model IIIP	hpIIIP
HP LaserJet Model IIID	hpIIID
HP LaserJet Model IV	hpIV
HP LaserJet Model 4M	hp4m
any remote printer	remote
default printer	unknown

You can set up other printers by using 'unknown' and then responding to the prompts, using values similar to those for supported printers.

Responding with 'remote' allows you to designate a remote system for printing. In this case, only four `printcap` file entries are required:

- `rm` (name of the remote system)
- `rp` (name of the printer on the remote system)
- `sd` (pathname of the spooling directory on this system)
- `lp` (the local line printer device, which is always null)

The `lp` parameter must be present to print to a remote printer.

Responding with 'printer?' allows you to enter a mode where more information can be requested for each printer type. In this mode you are prompted to enter the same printer types as listed in the previous table. Information about the printer and the default `printcap` file entries for that printer are displayed. Enter 'quit' to return to the prompt to select the printer type being added.

When specifying the printer type, you must use full command names and printer names. The default printer type is 'unknown'.

To install third-party printers, consult the documentation that came with the printer.

11.4.1.3 Printer Synonyms

The printer synonym is an alternate name for the printer. Some examples include 'draft', 'letter', and 'LA-75 Companion Printer'. You can enter as many alternate names for a printer as you like, but the total length of the line containing all the names must be less than 80 characters. When entering printer synonyms that can consist of many names, the entry process is terminated when you either enter a blank line or enter a line containing only white space.

After entering a synonym, you are prompted again. If you do not want to enter any more synonyms, press Return to continue.

Each synonym (including the printer number) identifies the printer to the print system. For example, if you chose the synonym 'draft' for a printer, the following command prints files on this printer:

```
$ lpr -Pdraft files
```

11.4.1.4 Device Special File

The device special file provides access to the port on the computer to which the printer is connected. The device special file is used if the printer is directly connected to a local serial or parallel port. In this case, you must equate a printer device logical name to the printer's device special file name by using the `lp` symbol in the `/etc/printcap` file. For example:

```
lp=/dev/lp
```

The installation procedure creates some device special files for the hardware that is connected to your computer. Usually, the device special files for parallel printers are named `/dev/lp`*n* (for example: `lp1`, `lp2`, `lp3`), and the device special files for serial line printers are named `/dev/tty`*nn* (for example: `tty00`, `tty01`, `tty02`). The *n* and *nn* variables specify the number of the printer.

When you use lprsetup, the program defaults to the next consecutive number when it sets up this file. For example, the default device pathname for the third serial line printer is /dev/tty03.

The default device special file is /dev/lp, which specifies a parallel printer.

For remote printers, you should specify a null argument with the lp symbol. For example:

```
lp=
```

Note

If the port is used for logins, the lprsetup script turns off the terminal line established by the getty process so the terminal line can be used for a printer.

11.4.1.5 Printer Accounting

The af parameter specifies the name of the accounting file used to keep track of the number of pages printed by each user for each printer. The name of the accounting file should be unique for each printer on your system. Use the pac program to summarize information stored in the printer accounting files. This file must be owned by user daemon and group daemon, which it will be if you use the lprsetup program to specify the printer accounting file.

The af parameter is not applicable for remote printer entries because the accounting policy for remote printers is employed at their (remote) systems.

Accounting is accomplished through programs called print filters. The lprsetup script does prompt you for the line print filter information. You must specify this information when you are prompted at the end of the lprsetup display for symbols to modify. Two print filter symbols, if and of, are needed for accounting. You modify them at this point. For example:

```
if=/usr/lbin/ln03rof
of=/usr/lbin/ln03rof
```

If you want to use separate accounting files for each printer on your system, the file names must be unique. However, an unlimited number of printers can share an accounting file. You cannot specify an accounting file for remote printers.

Accounting files must be owned by the print daemon. If you specify an accounting file, intermediate directories are automatically created as needed.

Note

Printer accounting does not work for PostScript files.

11.4.1.6 Spooler Directory

The `sd` parameter specifies the spooling directory where files are queued before they are printed. Each spooling directory should be unique. All `printcap` file entries must specify a spooling directory, both local and remote.

When the spooling directory is created, intermediate directories are created as necessary.

11.4.1.7 Error Log File

The `lf` parameter specifies the log file where errors are reported. The default log file, if one is not specified, is `/dev/console`. If you have more than one printer on your system, give each log file a unique name.

When the error log file is created, intermediate directories in the pathname are created as necessary.

11.4.1.8 Connection Type

The `ct` parameter specifies the type of connection to the printer. You can connect a printer directly to your computer from a port or terminal line. You can access networked printers that are connected to a LAT (Local Area Transport) terminal server or to a remote host. If you are using `lprsetup`, the choices for the connection type are:

- `dev` for local devices
- `LAT` for LAT devices (must be specified in uppercase)
- `remote` for remote devices

The `lprsetup` program supplies the default value `dev`.

11.4.1.9 Baud Rate

The baud rate is the maximum rate at which data can travel between the data source and the printer (for example, 4800 or 9600). The default baud rate for your printer should appear in the printer documentation. If you reset this baud rate yourself during the installation of the printer hardware, the rate that you set on the printer must match the rate that you enter in the `/etc/printcap` file.

You specify a baud rate only for serial printers that are directly connected to your computer. Baud rates are not specified for printers connected to the console port or connected by a parallel port or LAT port.

11.4.2 Using lprsetup to Install a Printer

This section describes how to install a printer locally (directly connected to your computer) using the `lprsetup` program. You can also use this program to modify a printer's configuration or remove a printer. These other tasks are described in Section 11.5.

Digital recommends that you accept the default values for an initial printer installation.

The printer described in the following example is an LN03R.

You can run the `lprsetup` program three ways:

- Select the Applications menu from a Motif window and choose System Setup from the menu
- Enter the `/usr/sbin/setup` command at the prompt and choose Printers from the menu
- Enter the `/usr/sbin/lprsetup` command at the prompt

You must have superuser privileges to run the `lprsetup` program. Depending on the type of printer you are adding and the information you provide, the `lprsetup` program might do the following:

- Create, or edit the existing `/etc/printcap` file
- Create a spooling directory
- Create an error log file
- Create an accounting file
- Create the device special files
- Prompt you to modify previously selected symbols

When you run the `lprsetup` script, the first display is the main menu:

```
# /usr/sbin/lprsetup
Digital UNIX Printer Setup Program

Command < add modify delete exit view quit help >:
```

The `lprsetup` command options are described in Table 11–2.

Table 11–2: lprsetup Options

Command	Description
`add`	Adds a printer
`modify`	Modifies an existing printer's characteristics
`delete`	Removes an existing printer from your configuration
`exit`	Exits from the `lprsetup` program
`view`	Displays the current `/etc/printcap` file entry for the printer you are configuring
`quit`	Exits from the `lprsetup` program
`help`	Displays online help about the `lprsetup` program

You can abbreviate any command option with its initial letter.

You can enter information at each prompt or press Return to select the default information provided. (In most instances, you can accept the defaults.) You can also enter a question mark (?) to get a description of the information you specify at the prompt.

Note

Some of the symbols displayed in the `lprsetup` script are not supported by the Digital UNIX operating system. Refer to Table 11–4 and to the `printcap`(4) reference page for information on the supported symbols.

The following example shows how to use the `lprsetup` command to set up an LN03R printer to be used by the local system:

```
# /usr/sbin/lprsetup
Digital UNIX Printer Setup Program

Command  < add modify delete exit view quit help >: add

Adding printer entry, type '?' for help.

Enter printer name to add [0] :  Return

For more information on the specific printer types
enter 'printer?'

Enter the FULL name of one of the following printer types:

la50   la70   la75   la324  la424  lg02   lg06   lg12
lg31   lj250  ln03   ln03s  ln05   ln06   ln07   ln08
lf01r  ln03r  ln05r  ln06r  ln07r  ln08r  ibmpro nec290
fx80   fx1050 hpIIP  hpIIIP hpIIID hpIV   hp4m   remote
unknown
```

```
or press RETURN for [unknown] : ln03r
Enter printer synonym: tomf
Enter printer synonym:  Return
Set device pathname 'lp' [] ? /dev/tty01
Do you want to capture print job accounting data ([y]|n)? y
Set accounting file 'af' [/usr/adm/lpacct]? Return
Set spooler directory 'sd' [/usr/spool/lpd] ?  Return
Set printer error log file 'lf' [/usr/adm/lperr] ?  Return
Set printer connection type 'ct' [dev] ?  Return
Set printer baud rate 'br' [4800] ? 9600
```

After you respond to each of the prompts, `lprsetup` prompts you to determine if you want to change any of the values assigned to the various symbols in your `/etc/printcap` file or if you want to specify any additional symbols. For example, you can set a specific page length or width. If you want to make any changes or add information, enter the appropriate symbol name. Refer to the `printcap`(4) reference page or Table 11–4 for more information on the various symbols.

```
Enter the name of the printcap symbol you wish to modify.  Other
valid entries are:
        'q'     to quit (no more changes)
        'p'     to print the symbols you have specified so far
        'l'     to list all of the possible symbols and defaults
The names of the printcap symbols are:

 af  br  cf  ct  df  dn  du  fc  ff  fo  fs  gf  ic  if  lf  lo
 lp  mc  mx  nc  nf  of  op  os  pl  pp  ps  pw  px  py  rf  rm
 rp  rs  rw  sb  sc  sd  sf  sh  st  tf  tr  ts  uv  vf  xc  xf
 xs  fo  ic  nc  ps  Da  Dl  It  Lf  Lu  Ml  Nu  Or  Ot  Ps Sd
 Si  Ss  Ul  Xf

Enter symbol name: q

        Printer #0
        ----------
Symbol  type  value
------  ----  -----
  af    STR    /usr/adm/lpacct
  br    INT    9600
  ct    STR    dev
  fc    INT    0177777
  fs    INT    03
  if    STR    /usr/lbin/ln03rof
  lf    STR    /usr/adm/lperr
  lp    STR    /dev/tty01
  mc    INT    20
  mx    INT    0
  of    STR    /usr/lbin/ln03rof
  pl    INT    66
  pw    INT    80
  rw    BOOL   on
```

```
  sd    STR    /usr/spool/lpd
  xc    INT    0177777
  xf    STR    /usr/lbin/xf
  xs    INT    044000
Are these the final values for printer 0 ? [y] y
```

Next, the lprsetup script prompts you to add comments to the /etc/printcap file. Enter n at the prompt if you do not want to add comments. Enter y at the prompt if you want to add comments. At the pound sign (#) prompt, enter your comment. Press Return at the pound sign (#) prompt to exit. The comments will be insert directly above the printcap entry in the /etc/printcap file.

```
Adding comments to printcap file for new printer, type '?' for help.
Do you want to add comments to the printcap file [n] ? : y
# Use this printer for draft-only
#  Return

Set up activity is complete for this printer.
Verify that the printer works properly by using
the lpr(1) command to send files to the printer.

Command  < add modify delete exit view quit help >: view

# Use this printer for draft-only
pearly|lp0|3X27|tomw:\
        :af=/usr/adm/lpacct:\
        :br#9600:\
        :ct=dev:\
        :fc#0177777:\
        :fs#03:\
        :if=/usr/lbin/ln03rof:\
        :lf=/usr/adm/lperr:\
        :lp=/dev/tty01:\
        :mc#20:\
        :mx#0:\
        :of=/usr/lbin/ln03rof:\
        :pl#66:\
        :pw#80:\
        :rw:\
        :sd=/usr/spool/lpd:\
        :xc#0177777:\
        :xf=/usr/lbin/xf:\
        :xs#044000:

Command  < add modify delete exit view quit help >: exit
```

Refer to the lprsetup(8) reference page for more information on using the program.

11.4.3 Setting Up Remote Printers

You can use the `lprsetup` script to set up remote printers that are accessible from a Local Area Transport (LAT) or from a remote machine.

If you are setting up a remote printer from a remote machine, the local machine (the client) must be listed in the `.rhosts` file of the remote machine (the host).

If your printer will be connected to a remote LAT terminal server, ensure that the LAT subsets are installed as described in the *Installation Guide*. To see if the LAT subsets are installed, enter:

```
# setld -i | grep OSFLAT
```

See the *Network Administration* guide for information on how to enable remote LAT terminal server printing.

11.4.4 Testing Printers

Test your printer by using the `lpr` command to print a few pages of text. You should also test any special printer features that you intend to use regularly on this printer, for example, PostScript or double-sided print. Refer to the `lpr`(1) reference page for more information on how to invoke these features. Table 11–4 lists the `printcap` symbol names, the type of values they take, default values, and descriptions.

The `lptest` command writes a traditional ripple test pattern to the standard output, or you can direct the output to a printer. A pattern that contains all 96 printable ASCII characters in each column is printed using 96 lines. In the pattern, each printed character is displaced rightward one character column on each successive line. This test is also useful for ascertaining the number of lines per page and the default page parameters. You can use the ripple test pattern to test printers, terminals, and drive terminal ports during debugging.

The `lptest` command has the following syntax:

/usr/sbin/lptest [*length* [*count*]]

Use the `lptest` command if you need quick output of random data. For example:

```
# /usr/sbin/lptest |lpr -P3r44
```

Refer to the `lptest`(8) reference page for more information.

11.5 Routine Operations

The first part of this chapter showed you how to set up the first printer on a system. This section describes the routine administrative tasks for a

print system that is already set up. You can use the `lprsetup` script to perform these tasks. You can also perform the tasks manually by editing system files and creating files and directories. If you are making many changes to the print system at one time, it may be easier for you to make the changes manually. The tasks described in the following sections are:

- Adding new printers to the system
- Modifying characteristics of existing printers
- Removing printers from the system
- Enabling printer accounting
- Controlling printer operations by using the `lpc` command

Note that if you manually remove printers from the `/etc/printcap` file, you also have to remove spooling, accounting, and error directories and files.

11.5.1 Adding Printers

Once you have one printer set up, you can add other printers at any time. Gather the following information about each printer:

- Printer name
- Printer type
- Printer synonyms
- Device pathname
- Accounting file name
- Spooling directory name
- Error log file name
- Connection type
- Baud rate

You can add printers by running the `lprsetup` command or you can add printers manually by performing the following steps:

1. Create a printer spooling directory. Refer to Section 11.6.2.2.
2. Create the `/etc/printcap` file and edit it to include a description of the printer. Refer to Section 11.6.3.
3. Create an accounting file and a log file and enable printer accounting. Refer to Section 11.5.4.

You should make sure that the `/etc/inittab` file does not invoke the `getty` process on serial lines that have printers attached. If you use the `lprsetup` script, this is done for you.

11.5.2 Modifying Printers

To modify a printer's configuration automatically, run the `lprsetup` program and choose the modify option from the main menu. Section 11.4.2 describes how to use the `lprsetup` program.

If you change the name of the spooling directory, the accounting file, or the error log file, `lprsetup` asks you to verify that the information is correct before it deletes the original information.

To manually modify a printer's configuration, edit the `/etc/printcap` file and modify the entry that pertains to the printer. Refer to Section 11.6.3 and to the `printcap`(4) reference page for information about the `/etc/printcap` file symbols.

11.5.3 Removing Printers

To remove a printer, run the `lprsetup` program and choose the delete option from the main menu. The program prompts you for the printer name. Enter the name of the printer you want to remove. You are prompted for confirmation that you want to delete the error log file and the accounting file because these files can be shared by more than one printer. If you do have shared files, do not delete them. Section 11.4.2 describes how to use the `lprsetup` program.

If you have included comments for the printer in the first line of its `/etc/printcap` file entry, the `lprsetup` program does not delete them. You must edit the `/etc/printcap` file and delete the comments.

To manually remove a printer, edit the `/etc/printcap` file and delete the entry that pertains to the printer. You must also manually delete the accounting and log file and the spooling directory.

11.5.4 Enabling Printer Accounting

Printer accounting allows you to charge users for printing services and to determine the amount of printer usage.

There are two types of printer accounting: printer user accounting and printer summary accounting. Printer user accounting provides information about printer use according to the machine and user name that issues the print request. Printer summary accounting provides information about the amount of media (number of printed pages or number of feet of roll paper or film) the printer produces. You specify the `pac` command with the `–s` option to produce printer summary accounting information.

The printer accounting files default to the `/var/adm` directory. If you use the `lprsetup` program to add a printer, the program creates the accounting

file you specify. The /usr/adm/lpacct file is the default accounting file. If you add a printer manually, you must create the accounting file.

Note

The /var/adm/printer directory should be owned by user adm and belong to group adm. The printer accounting files should have protection mode 644, be owned by user adm, and belong to group system.

Refer to Chapter 13 for more information on using printer accounting.

11.5.5 Controlling Local Print Jobs and Queues

Use the lpc command to manage the print jobs and queues associated with the local printers on your system. You can use the lpc command to:

- Enable and disable printers and spooling queues
- Change the order of queued jobs
- Display the status of the printer, queue, and daemon

Some lpc commands, for example, the disable command, require you to be superuser.

Note

You can use the lpc command only to manage print queues that are local to your system. Although a remote printer has both a local queue and a remote queue, the lpc command manages only the local queue.

There are 15 command arguments that you can specify with the lpc command. You can also use the lpc command interactively. If you enter the lpc command without any command arguments, the lpc> prompt is displayed. You can then enter command arguments.

The lpc command has the following syntax:

/usr/sbin/lpc [*argument*] [all | *printer*...]

Some of the command arguments allow you to specify all to indicate all the printers or to specify one or more *printer* variables to indicate a specific printer.

You can specify the *argument* variables defined in Table 11–3.

Table 11–3: lpc Command Arguments

lpc Argument	Description
help [*argument*]	Prints a one-line description of the specified lpc command argument. If an *argument* variable is not specified, the list of arguments is displayed.
? [*argument*]	Same as the help argument.
abort	Terminates an active lpd daemon and then disables printing. This prevents the lpr or lp command from starting a new lpd daemon.
clean	Removes any temporary files, data files, and control files that cannot be printed (for example, files that do not form a complete printer job) from the specified print spooling directory.
disable	Turns off the specified print spooling queue. This prevents the lpr or lp command from entering new jobs in the queue.
down *message...*	Turns off the specified print queue, disables printing, and enters the specified message in the printer status file. The message does not need to be quoted because remaining arguments are treated the same as echo. You can use the down argument to take down a printer and inform users. If a printer is down, the lpq command indicates that the printer is down.
enable	Enables spooling for the specified printers. This enables the lpr or the lp command to enter print jobs in the spooling queue.
exit	Exits from lpc.
quit	Exits from lpc.
restart	Attempts to start a new lpd daemon for the specified printer. This argument is useful if some abnormal condition causes the daemon to terminate unexpectedly and leave jobs in the queue. If this occurs, the lpq command indicates that no daemon is present. If a daemon is hung, you must first kill the process and then restart the daemon by using the restart argument.
start	Enables printing and starts a spooling daemon for the specified printer.
status [*printer*]	Displays the status of the specified printer daemon and queue. The status argument shows if the queue is enabled, if printing is enabled, the number of entries in the queue, and the status of the printer's lpd daemon. If a printer name is not supplied, information about all printer daemons and queues is displayed.

Table 11–3: lpc Command Arguments (cont.)

lpc Argument	Description
stop	Stops a spooling daemon after the current job is complete and disables printing.
topq *printer*	Puts print jobs in the queue in the specified order. You can specify the print jobs by also specifying a *request_ID* variable or a *username* variable.
up	Enables all printing and starts a new printer daemon. Cancels the down argument.

The following example shows that the lpd daemon is active on the printer named tester and there is one entry in the queue:

```
# /usr/sbin/lpc
lpc> status tester
tester:
        printer is on device '/dev/tty02' speed 9600
        queuing is enabled
        printing is enabled
        1 entry in spool area
lpc>
```

Refer to the lpc(8) reference page for more information.

11.6 Reference Information

This section contains information about the line printer daemon, lpd, and the system files that are required for print system operations. These files are created automatically if you use the lprsetup script, as described in Section 11.4.2, or you can create and modify the files manually. Note that if you create files manually, you will also need to manually change the /etc/printcap file, so the changes can take effect.

11.6.1 Line Printer Daemon

Printers are controlled by the line printer daemon, lpd, which is located in the /usr/lbin directory. Printing cannot take place unless the lpd daemon is running. The lpd daemon has many functions:

- Handles printer spooling, which is the mechanism by which a file is placed in a queue until the printer can print the file.
- Uses the listen and accept system calls to control printers and to ensure that the user who requested printing is allowed to use the printer.

- Scans the /etc/printcap file to determine printer characteristics.
- Uses specific print filters for print requests. Print filters translate an input format into a printer-specific output format.
- After a system reboot, prints any files that were not printed when the system stopped operating.

When you use the lpr command, the lpd daemon is activated, and the daemon copies the file to the printer's spooling queue or directory. Requests are printed in the order in which they enter the queue. A copy of the file to be printed remains in the queue until the printer is ready to print it; then the lpd daemon removes the file from the spooling queue and sends it to the printer.

After you install and boot your system, the lpd daemon is usually started by the init program. You can start the lpd daemon with the following command syntax:

/usr/lbin/lpd [-l]

The –l option causes the lpd daemon to log valid requests from the network. This option is useful for debugging.

To test whether the line printer daemon is running, enter:

```
# ps agx |grep /usr/sbin/lpd
```

11.6.2 Spooling Directories

Each printer must have its own spooling directory located under the /usr/spool/lpd directory. The spooling directory acts as a printer's spooling queue; it contains the files that are queued for printing on that printer. A printer spooling directory should have the same name as the printer reference name and must be located on the machine attached to the printer. The printer reference name is the name that you specify to print on a particular printer.

If you are using lprsetup, the program supplies the default value /usr/spool/lpd*n*. The *n* variable specifies the printer number. For example, the default name of the spooling directory for a second line printer could be /usr/spool/lpd2. The default spooling directory for any printer is /usr/spool/lpd.

Each printer entry in the /etc/printcap file should specify a spooling directory even if the printer is connected to another machine or is on another network. You specify a spooling directory with the sd symbol. For example:

```
sd=/usr/spool/lpd/purple
```

Spooling directories must have the same parent directory name, which is normally /usr/spool.

11.6.2.1 Spooling Directory Files

A spooling directory contains a status file and a lock file that are created by the lpd daemon when a file is queued for printing. The lock file contains control information about the current print process. For example, it can inform the lpd daemon that the printer is printing a job. The lock file prevents the lpd daemon from invoking another job on the printer while a file is printing. The lock file contains the process identification number of the daemon that is currently running. The status file contains a line that describes the current printer status. This line is displayed if a user inquires about printer status. If a printer whose status is queried is not active, the status message written to standard output is no entries.

When the lpd daemon is activated as a result of a print request, it looks in the printer spooling directory for a lock file. If a lock file is not found, the lpd daemon creates one and writes the identification number and the control file name on two successive lines in the file. The lpd daemon then scans the printer spooling directory for command files whose names begin with cf. Command files specify the names of user files to be printed and contain printing instructions for the files. Each line in a command file begins with a key character that indicates what to do with the remainder of the line. The key characters and their meanings are described in detail in the lpd(8) reference page.

Data files, whose names begin with df, are also located in the spooling directory. Data files contain text formated for printing. These files are identified by their print request identification numbers only.

After a file is printed, the lpd daemon removes the control and data files from the printer spooling queue, updates the status file, and sets up the next file in the spooling queue for printing.

For example, if a printer named milhaus has jobs currently waiting to be printed, the following command lists the files that are stored in the spooling directory:

```
# ls –l /var/spool/lpd/milhaus
-rw-rw----   1    root      75  Jan 17  09:57  cfA0220mothra
-rw-rw----   1    root      96  Jan 17  10:03  cfA143harald
-rw-rw----   1    root  199719  Jan 17  09:57  dfA0220mothra
-rw-rw----   1    root    9489  Jan 17  10:03  dfA143harald
-rw-r--r--   1    root      20  Jan 17  10:06  lock
-rw-rw-rw-   1  daemon     113  Jan 17  10:00  status
```

11.6.2.2 Creating a Spooling Directory

If you want to manually add a printer, use the `mkdir` command to create the spooling directories for each printer. The spooling directory permission mode must be set to 775. The directory's group and ownership must be set to the name `daemon`. For example:

```
# cd  /var/spool/lpd
# mkdir lp1
# chmod 775 lp1
# chgrp daemon lp1
# chown daemon lp1
# ls -l lp1
drwxr-xr-x  2  daemon  daemon  24  Jan 12  1994  lp1
```

11.6.3 The /etc/printcap File

The `lpd` daemon uses the `/etc/printcap` printer database file to print requests. Each entry in the file describes a printer. Printer characteristics are specified by 2-letter abbreviations called symbols. The symbols are described in this section and in the `printcap`(4) reference page. The `lprsetup` program modifies the `/etc/printcap` file.

The following example shows an `/etc/printcap` entry for both a local printer and a remote printer:

```
#
#
lp|lp0|0|dotmatrix|mary:\
      :af=/usr/adm/printer/lp.acct:\
      :br#9600:\
      :ct=dev:\
      :fc#0177777:\
      :fs#023:\
      :if=/usr/lbin/la75of:\
      :lf=/usr/adm/lperr:\    1
      :lp=/dev/tty01:\
      :mx#0:\
      :of=/usr/lbin/la75of:\
      :pl#66:\
      :pw#80:\
      :sb:\     2
      :sd=/usr/spool/lpd:\
      :xc#0177777:\     3
      :xf=/usr/lbin/xf:\
      :xs#044000:\
#
#
sqirrl|3r3|ln03r3|postscript3|In office 2T20:\
```

```
        :lp=:rm=uptown:rp=lp:sd=/var/spool/printer/ln03r3:mx#0:\ 4
#
```

The callouts in the /etc/printcap entry show the following possible symbol syntaxes:

1 Specifies a symbol with alphabetic characters.

2 Specifies a symbol that represents a Boolean expression.

3 Specifies a symbol with a numeric value.

4 Specifies an entry for a remote printer. The lp, rm, rp, and sd symbols are required for remote printers for which you are a client.

The first line of a printer entry contains the fields that specify the printer primary reference name and printer name synonyms. This first line and these fields are required for every printer, both local and remote.

The printer reference name is the name that you subsequently use in order to specify printing to this printer. You can give each printer as many alternative reference names as you want, but each field on the first line must be separated with a vertical bar (|). The first line must end with a colon (:).

Note

A local printer entry in the /etc/printcap file should have the default printer reference name lp0 so that print jobs can have a destination when printer reference names are not specified in print commands.

The remaining lines of each printer entry contain the descriptive symbols and values that define the printer's configuration. Symbols are 2-character mnemonics and can be specified with an equal sign (=) and alphabetic characters or with a number sign (#) and a numeric value. Some symbol names have Boolean equivalents, which do not use parameters. You can specify the symbols on one line or on individual lines, but you must separate them with colons (:).

To make the /etc/printcap file easy to read, you can place a colon (:) at the beginning of a line and a backslash (/\) at the end of a line to separate the symbols.

Table 11–4 lists the printcap symbol names, the type of values they take, default values, and descriptions.

Table 11–4: The printcap File Symbols

Symbol	Type	Default	Description
af	alphabetic	NULL	Name of accounting file
br	numeric	none	If lp is a tty, set the baud rate (ioctl call)
cf	alphabetic	NULL	The cifplot data output filter
ct	alphabetic	NULL	Connection type[a]
df	alphabetic	NULL	The TeX data filter (DVI format)
dn	alphabetic	/usr/lbin/lpd	Specifies a nonstandard daemon pathname
du	numeric	none	Specifies a nonstandard daemon UID
fc	numeric	0	If lp is a tty, clear flag bits (`sgtty.h`)
ff	alphabetic	/f	String to send for a form feed
fo	Boolean	false	Print a form feed when device is opened (to suppress all form feeds, specify both the fo and sf symbols)
fs	numeric	0	If lp is a tty, set flag bits
gf	alphabetic	NULL	Graph data filter (plot format)
ic	Boolean	false	Driver supports (nonstandard) ioctl to independent printout
if	alphabetic	NULL	Accounting text filter
lf	alphabetic	/dev/console	Error log file name
lo	alphabetic	lock	Name of lock file
lp	alphabetic	/dev/lp	Printer device logical name[b]
mc	numeric	20	Specifies the maximum number of copies allowed
mx	numeric	1000	Maximum file size (in BUFSIZ blocks), zero (0) removes size restriction
nf	alphabetic	NULL	The ditroff data filter (device independent troff)
of	alphabetic	NULL	Output filtering program
op	alphabetic	NULL	Entry in the reference name field for LAT port characteristics

[a]The value for the `ct=` type is either `dev` or `LAT`. When the printer is connected directly to a local port, a device connection is used. You define a printer device special file name for the `lp` symbol.
[b]When the printer is connected to a remote LAT printer, the `lp` symbol must specify a configured LAT application port. Refer to the *Network Administration* guide for information on setting up a LAT, configuring a LAT application port, and enabling remote LAT terminal server printing.

Table 11–4: The printcap File Symbols (cont.)

Table 11–5: The printcap File Symbols, continued

Symbol	Type	Default	Description
os	alphabetic	NULL	Service name supported on some terminal servers
pl	numeric	66	Page length (in lines)
pp	alphabetic	NULL	The print command filter replacement
ps	alphabetic	non_PS	Indicates that the printer is PostScript
pw	numeric	132	Page width (in characters)
px	numeric	0	Page width in pixels (horizontal)
py	numeric	0	Page length in pixels (vertical)
rf	alphabetic	NULL	The FORTRAN-style text file filter
rm	alphabetic	NULL	Machine name for remote printer[a]
rp	alphabetic	lp	Remote printer name argument
rs	Boolean	false	Restrict remote users to those with local accounts
rw	Boolean	false	Open the printer device for reading and writing
sb	Boolean	false	One-line banner
sc	Boolean	false	Suppress multiple copies
sd	alphabetic	/usr/spool/lpd	Spool directory
sf	Boolean	false	Suppress all form feeds, except those that are in the file
sh	Boolean	false	Suppress printing of burst page header
st	alphabetic	status	The status file name
tf	alphabetic	NULL	The troff data filter (catphototypesetter)
tr	alphabetic	NULL	Print trailing string if queue empties (the trailing string can be a series of form feeds or an escape sequence)
ts	alphabetic	NULL	LAT terminal server node name.

Table 11–5: The printcap File Symbols, continued (cont.)

Symbol	Type	Default	Description
vf	alphabetic	NULL	The raster image filter (you can also specify raster filters with the if and of symbols)
xc	numeric	0	If lp is a tty, clear local mode bits (tty)
xs	numeric	0	If lp is a tty, set local mode bits

[a]If the printer is a remote printer, a remote connection is used. You must use the `rm` symbol to specify the name of the machine to which the printer is attached. You must also specify the printer reference name with the `rp` symbol, as well as the `lp` and `sd` symbols

11.6.4 Line Printer Daemon Filter Directory

The filter directory for the `lpd` daemon translates data that you want to print into the format appropriate for your printer. You must specify the filter that matches each printer on your system. For example, to print files with the LN03R printer, you would use the `ln03rof` filter.

You can specify an accounting filter with the `if` symbols and an output filter with the `of` symbol. Output filters filter text data to the printer device if accounting is not enabled or if text data must be passed through a filter. For example:

```
if=/usr/lbin/ln03rof
of=/usr/lbin/ln03rof
```

Refer to the `lpd`(8) reference page for more information on using filter capabilities.

Table 11–6 lists the available print filters located in the `/usr/lbin` directory.

Table 11–6: Print Filters

Filter Name	Description
`epsonof`	Epson FX-80 and FX-1050 printers
`hplaserof`	NEC Silentwriter Model 290 and most Hewlett Packard LaserJets
`hplaserpsof`	Hewlett Packard PostScript-equipped LaserJets
`hplaser4psof`	Hewlett Packard LaserJet Model 4M only
`lpf`	Line printer filter (LP25, LP26, LP27, LP29, LG01, LA210, LQP02, LQP03)

Table 11–6: Print Filters (cont.)

Filter Name	Description
lqf	Letter-quality filter (LQP02, LQP03)
la75of	Dot matrix printer filter (LA75, LA70, LA324, LA424)
lg31of	LG31 line printer filter
lg02of	LG02 ink jet printer filter
ln03of	LN03 laser printer filter
ln03rof	LN03 ScriptPrinter filter
ln03rof_isolatin1	LN03 ScriptPrinter filter with ISO Latin_1 encoding
ln03rof_decmcs	LN03 ScriptPrinter filter with DEC multinational character set encoding
ln05of	DEClaser 2100 laser printer filter
ln05rof	DEClaser 2150 PostScript printer filter
ln05rof_isolatin1	DEClaser 2150 PostScript printer filter with ISO Latin_1 encoding
ln05rof_decmcs	DEClaser 2150 PostScript printer filter with DEC multinational character set encoding

Table 11–7: Print Filters, continued

Filter Name	Description
ln06of	DEClaser 2200 laser printer filter
ln06rof	DEClaser 2250 PostScript printer filter
ln06rof_isolatin1	DEClaser 2250 PostScript printer filter with ISO Latin_1 encoding
ln06rof_decmcs	DEClaser 2250 PostScript printer filter with DEC multinational character set encoding
ln07of	DEClaser 1100 laser printer filter
ln07rof	DEClaser 1150 PostScript printer filter
ln07rof_isolatin1	DEClaser 1150 PostScript printer filter with ISO Latin_1 encoding
ln07rof_decmcs	DEClaser 1150 PostScript printer filter with DEC multinational character set encoding
ln08of	DEClaser 3200 laser printer filter
ln08rof	DEClaser 3250 PostScript printer filter

Table 11–7: Print Filters, continued (cont.)

Filter Name	Description
ln08rof_isolatin1	DEClaser 3250 PostScript printer filter with ISO Latin_1 encoding
ln08rof_decmcs	DEClaser 3250 PostScript printer filter with DEC multinational character set encoding
ln09of	DEClaser 5100 PostScript printer filter
ln09of_isolatin1	DEClaser 5100 PostScript printer with ISO Latin_1 encoding
ln09of_decmcc	DEClaser 5100 PostScript printer with DEC multinational character set encoding
lj250of	LJ250 DEColorwriter filter

For printers not supplied by Digital, consult your printer manual for filter information.

11.6.5 Flag Bits

Flag bits specify characteristics about data transmission from the host to the printer and, if possible, from the printer to the host on a serial line only (LAT and RS232). Data that is passed from the printer to the host may include stop and start status information, which tells the host that the printer input buffer can accept input or that it is about to overflow.

Delays are specific times used to slow the transmission of the next group of characters to the input buffer. Delays give the printer mechanism time to perform operations such as a carriage return, newline, tab, and form feed.

Flag bits are cleared with the `fc` symbol and set with the `fs` symbol. All printers do not use all the flag bits, but you must either set the bits or clear them. You should consult your printer manual for specific information about flag bits.

The flag bits are specified as octal numbers in a 16-bit word. Octal values are preceded with the number zero (0). To clear all the bits, specify the value 0177777 with the `fc` symbol. To set all the bits, specify the value 0177777 with the `fs` symbol. All bits should be cleared (using `fc#0177777`) before calling the `fs` symbol. To set or clear any groups of bits, specify the octal sum of the combined bits for the number of flag bits.

The following is an example of flag bit specifications:

```
fc#0177777
fs#0141
```

As shown in the previous example, `fc#0177777` clears all bits. The `fs` symbol set to `0141` specifies the `OPOST`, `ONLRET`, and `OFILL` flag bits.

Table 11–8 lists each flag bit name, its octal value, and its description.

Table 11–8: Flag Bits

Flag	Octal Value	Description
OPOST	0000001	Enable output processing
ONLCR	0000002	Map NL to CR-NL
OLCUC	0000004	Map lower case to upper case
OCRNL	0000010	Map CR to NL
ONOCR	0000020	No CR output at column 0
ONLRET	0000040	NL performs CR function
OFILL	0000100	Use fill characters for delay
OFDEL	0000200	Fill is DEL, else NUL
NLDLY	0001400	Newline delay
NL0	0000000	
NL1	0000400	
NL2	0001000	
NL3	0001400	
TABDLY	0006000	Horizontal tab delay
TAB0	0000000	
TAB1	0002000	
TAB2	0004000	
TAB4	0006000	
CRDLY	0030000	Carriage Return delay
CR0	0000000	
CR1	0010000	
CR2	0020000	
CR3	0030000	
FFDLY	0040000	Form feed delay
FF0	0000000	
FF1	0040000	

Table 11–8: Flag Bits (cont.)

Flag	Octal Value	Description
BSDLY	0100000	Backspace delay
BS0	0000000	
BS1	0100000	
OXTABS	1000000	Expand tabs to spaces

Refer to the `tty`(7) reference page for detailed information on flag bits.

11.6.6 Mode Bits

Mode bits specify details about the capability of a particular terminal and usually do not affect printer operation. Mode bits are cleared with the `xc` symbol and set with the `xs` symbol. Some printers use all of the mode bits, so you must either set them or clear them. The mode bits are specified as octal numbers in a 16-bit word format. You should clear all bits by specifying `xc#0177777` before you specify the `xs` symbol.

Refer to the `tty`(7) reference page for a detailed description of the status bits.

The following is an example of mode bits specifications:

```
xc#0177777
xs#044000
```

As shown in the previous example, `xc#0177777` clears all bits. The `xs` symbol set to `0110` specifies the `ECHO` and `ECHOCTL` mode bits.

Table 11–9 lists a description of each mode bit.

Table 11–9: Mode Bits

Mode	Octal Value	Description
ECHOKE	0000001	Echos KILL by erasing the line
ECHOE	0000002	Visually erase characters
ECHOK	0000004	Echoes NL after KILL
ECHO	0000010	Enable echoing
ECHONL	0000020	Echoes NL even if ECHO is off
ECHOPRT	0000040	Echo erased chars between and /
ECHOCTL	0000100	Echo control characters as ^(char)

Table 11–9: Mode Bits (cont.)

Mode	Octal Value	Description
ISIG	0000200	Enable special chars INTR, QUIT and SUSP
ICANON	0000400	Enable canonical input
ALTWERASE	0001000	Use alternate word erase algorithm
IEXTEN	0002000	Enable FLUSHO and LNEXT
XCASE	0040000	Canonical upper/lower presentation

11.6.7 Remote Printer Characteristics

If a printer will be used by users on remote machines, `/etc/printcap` files on the local machine attached to the printer and on the remote machines that will use the printer must contain some network configuration information.

On the local machine attached to the printer you must specify the `rs` symbol, which specifies a Boolean value that takes only a true (yes) or false (no) value, along with the other printer configuration symbols. If you define the value as true, remote users must have an account on the local machine that is attached to the printer. If you define the value as false, remote users can access the local printer if the local printer is listed in the `.hosts` file. Refer to Section 11.6.3 for an example of an `/etc/printcap` file.

On the remote machine that will use the printer, you must specify the `rm`, `rp`, `lp`, and `sd` symbols.

The `rm` symbol specifies the name of the machine attached to the printer. For example:

```
rm=deccom
```

The `rp` symbol specifies the printer spool name on the remote system. For example:

```
rp=ln03lab
```

For remote printers, you should specify the `lp` symbol without a value:

```
lp=
```

The `sd` symbol specifies the spooling directory. For example:

```
sd=/usr/spool/lpd
```

11.6.8 Pagination and Imaging Parameters

Printer filters must know the size of an output page to perform proper page framing and line-feed and carriage returns (line folding).

For line printers, the `pl` and `pw` parameters specify the page length in number of lines (default is 66) and the column width in number of constant-width characters (default is 132), respectively. For example:

```
pl#55
pw#70
```

You should not specify a width of more than 80 characters for a letter-quality printer that uses 8 1/2-inch by 11-inch paper. If you specify a width that is greater than 80 characters on a printer, the page prints in landscape mode.

For high-resolution laser-type printers, the line length and page width parameters are `py` and `px`, which specify the number of pixels along the y- and x-coordinate planes of the printer output image area. Some printers can operate in either constant-width or imaging modes, so you must specify both sets of parameters. For example:

```
px#60
py#80
```

Refer to your printer's manual for its output characteristics.

11.7 Troubleshooting

This section provides a checklist for diagnosing printer problems. It also describes how print errors are logged in the `/usr/adm/lperr` file, providing this feature is specified in the `/etc/printcap` file.

11.7.1 Installation and Routine Operations

If a problem occurs on an existing printer or when adding a printer to a system, diagnose the problem as follows:

- Verify that the printer hardware is correctly installed and operating as expected. Most printers have internal test and print test options.
- Ensure that the correct settings are recorded in the `/etc/printcap` file. Refer to Section 11.6.3.
- Ensure that the printer daemon is present by using the following command:

  ```
  # ps agx |grep /usr/sbin/lpd
  ```

If the daemon is not running, restart it by using the following commands:

```
# rm -f /dev/printer /var/spool/lpd.lock
# /usr/lbin/lpd -l
```

The first command removes the /dev/printer and /var/spool/lpd.lock files. In the second command, the –l option causes the daemon to log requests from the network. This flag is useful for debugging problems with remote printers.

- Use the lpc command to check on the status of the printer. If queues are stalled, try resetting the queues (refer to Section 11.5.5).
- Ensure that the appropriate spooling or device files have been created and that ownership and access are correct (refer to Section 11.6.2).

11.7.2 Printer Error Logging

The lpd daemon logs printer errors to the error log file. Specifying an error log file is optional. If you used lprsetup to install the printer, the program provides the default value /usr/adm/lperr. If you do not specify an error log file, errors are logged to /dev/console.

The error log file is specified with the lf symbol in the /etc/printcap file. For example:

```
lf=/var/adm/lpderrs
```

Error log files are usually located in the /var/adm directory. An error log file can be shared by all local printers, but you should specify the file in each /etc/printcap file printer entry.

11.8 TCP/IP (telnet) Printing

TCP/IP printing, also called telnet printing, allows you to submit print jobs to a remote printer that is directly connected to the network. Note that to use this feature, your printer must contain a TCP/IP interface card and must be registered with a TCP/IP node name and node address.

With TCP/IP printing, the local host manages print jobs in the same manner as it would manage print jobs for a local printer. The only difference is that with TCP/IP printing, the local print daemon (lpd) communicates with the remote printer over TCP/IP (similar to LAT printing). Each printer listens for connection requests on a socket number that is specified in the printer hardware or that is user-defined through the printer console.

Although multiple hosts can talk to a single printer connected to the network in this way, the hosts are handled on a first-come, first-served

basis. Therefore, TCP/IP printing is not the same as remote printing, in which the remote printer manages a print queue on the remote site and listens for network connections on socket 515 (as specified in the entry for `printer` in `/etc/services`).

11.8.1 Setting up TCP/IP Printing

The following steps describe how to set up TCP/IP printing on a local host.

1. Set up the printer. Assign a TCP/IP address and node name to each printer with a network card. Also, determine the TCP/IP socket number on which the printer will listen for connection requests. You will need the socket number in Step 2b when you edit the `/etc/services` file. Table 11–10 lists the socket numbers for three printers made by Digital and one made by Hewlett Packard.

   Table 11–10: TCP/IP Socket Numbers

Printer	Socket Number
DEClaser 3500 (LN14)	10001
DEClaser 5100 (LN09)	10001
HP Laserjet 4m+	9100
LN17	2501

   To obtain the socket number for other printers, see your printer documentation. Some printers may allow you to specify this number yourself.

2. Configure the local host This step describes the utilities that you need to run and the files that you need to modify on the local host in order to configure TCP/IP printing. You must have superuser privileges to perform the following tasks:

   a. Configure the printer using `lprsetup` Execute the `/usr/bin/lprsetup` command and answer the questions to create an entry in the `/etc/printcap` file for your printer. When it prompts you to enter values for printcap control variables, assign the following values to the `ct` and `lp` variables:

      ```
      ct=tcp
      lp=@nodename/servicename
      ```

      Replace *nodename* with the name of the printer's node as registered for use on your network and replace *servicename* with the name you will choose to enter in the `/etc/services` database in the next step. If you want to modify an existing

/etc/printcap printer entry to use TCP/IP printing, edit the /etc/printcap file using a text editor, such as vi, and modify the values for the ct and lp variables. You can also remove the values for the xs, xc, fs, and fc control variables which establish settings that are relevant to the serial port driver. These are ignored by the network socket driver.

b. Configure the services database. You must register a service name and tcp port number (socket number) in the /etc/services database file. Enter the socket number that you determined when you configured the printer in step 1 and associate it with a service name of your choice. For example, to configure the services database for a DEClaser 3500, you would add the following line to the /etc/services file:

```
declaser3500    10001/tcp
```

Note that the user-defined declaser3500 string represents the service; it is the same string that you would have entered as the *servicename* in the /etc/printcap file in step 2a. After saving the changes to the /etc/services file, restart the inetd daemon to reload the /etc/services file with the printer information you just added. To do this, type the following command:

```
# rcinet restart
```

This stops and restarts the Internet network services on your system.

c. Configure the remote hosts database. The *nodename* value that you specified as part of the lp variable value in the /etc/printcap file must be known by your local host's network management services; therefore, you must enter the *nodename* and its network address in the /etc/hosts database file. If you are running a BIND server for remote host names, you do not necessarily need to add the printer's node name to the /etc/hosts file, though if there is ever a problem with the BIND server, an entry in /etc/hosts would be a useful fallback.

11.8.2 Using TCP/IP Printing

Once configured, TCP/IP printing is used like local and remote printing. From the command line, execute the lpr command specifying the node name of the printer, command options, and file names.

11.8.3 Known Restrictions on the Use of TCP/IP Printing

TCP/IP printing works when printing within a local subnet; however, printing in complex networks across one or more routers may cause reliability problems.

In addition, printing non-PostScript files with some PostScript and non-PostScript filters may yield unexpected results. Table 11–11 lists the filters with which you could experience these problems.

Table 11–11: Non-PostScript and PostScript Filters

Filter Name	Filter Type
lpf	Non-PostScript
la75of	Non-PostScript
la324of	Non-PostScript
lqf	Non-PostScript
hplaserof	PostScript

To provide expected behavior with older printers, these non-PostScript filters maintain a dependence on the serial port driver to automatically supply carriage returns after line feeds when you specify the (octal) 020 bit to the `fs` control variable in the `/etc/printcap` file.

Because this control bit is not interpreted by the network socket driver, the formatting behavior that would be supplied by the serial port driver is absent. Therefore, non-PostScript files that are not preformatted for the printer may not print out as they would in serial-port-connected configurations. In particular, this may affect ASCII text files that do not contain embedded carriage-returns.

Most printers using the `lpf`, `la75of`, `la324of`, and `lqf` non-PostScript filters do not provide network interface card support. However, the printing problems may still be an issue for users who use serial-and-parallel-port to network-port converters, like the Digital RapidPrint network interface box, which allow these printers to act like TCP/IP printers with built-in network support.

The `hplaser4psof` PostScript filter works for PostScript files and for preformatted non-Postscript files (like PCL files), but it will likely produce unexpected results for files that have not been preformatted (such as ASCII text without embedded carriage-returns).

12

Administering the Archiving Services

One of the more common tasks of a system administrator is helping users recover lost or corrupted files. To perform that task effectively, you must set up procedures for backing up files at frequent and regular intervals. This chapter discusses how to back up and restore files, and describes the following tasks:

- Backing up data
  - Choosing a backup schedule
  - Performing a full backup
  - Performing an incremental backup
  - Performing a remote backup
  - Using backup scripts
- Restoring data
  - Restoring a file system
  - Restoring a file system on a new partition
  - Restoring files
  - Restoring files interactively
  - Performing remote restores

Note

You can also use the *SysMan* `dxarchiver` command to perform some of these tasks.

The Digital UNIX operating system has other facilities you can use to back up and restore files:

- NetWorker SingleServer Save and Restore

  A component of Digital UNIX, used for single system archiving, and packaged as an optional subset to the operating system
- POLYCENTER NetWorker Save and Restore

A separately licensed product, used for networked systems archiving

- The `btcreate` and `btextract` utilities

  You can use these utilities to create and restore standalone, bootable tapes of the operating system and file systems.

These capabilities are briefly surveyed in the following sections. Then, the archiving tasks are described.

12.1 NetWorker SingleServer Save and Restore

NetWorker SingleServer Save and Restore is a graphical utility that backs up and recovers local files on a single machine to a local tape or loader.

Instructions for installing NetWorker SingleServer Save and Restore are in the Digital UNIX *Installation Guide*.

NetWorker SingleServer is a subset of Digital's POLYCENTER NetWorker Save and Restore which backs up and restores files on multiple machines across a TCP/IP network, in addition to single machines.

NetWorker SingleServer is shipped with preconfigured settings that provide you with the ability to start backing up files immediately.

NetWorker SingleServer protects your data by automating the day-to-day process of backing up the server. NetWorker SingleServer offers features similar to the multiclient NetWorker products – except that it supports a single client rather than multiple clients. When NetWorker SingleServer is installed on a machine, the machine becomes a client of itself.

Currently, a Digital UNIX user must know what utility (`tar`, `cpio`, `dump/restore`, or `vdump/vrestore`) was used to perform a backup in order to restore information from the archive. NetWorker SingleServer eliminates the need to know what utility was used to perform the backup.

The NetWorker SingleServer utility offers the following features:

- Preconfigured settings
- Label templates for electronically labeling tapes
- Ability to perform unattended backups
- Five backup schedules
- Five preconfigured policies for managing backed-up files
- Two preconfigured directives that assist you in streamlining backups
- Notification of NetWorker activity
- Easy recovery of files

12.2 POLYCENTER NetWorker Save and Restore

Digital offers POLYCENTER NetWorker Save and Restore for backing up a network of systems. Depending on your needs, this product will back up just a few systems or hundreds of systems on a single network. For more information, see your Digital representative.

12.3 Bootable Tape

You can create a bootable Standalone System (SAS) kernel on tape. The SAS kernel has a built-in memory file system (mfs), which contains the minimum commands, files, and directories needed to restore the system image. This is referred to as the miniroot file system.

To create the SAS kernel, you must use the `btcreate` utility. Once you have created the kernel, you can restore the customized image using the `btextract` utility. The following sections provide an overview of the `btcreate` and `btextract` utilities. For information on syntax and examples, see the reference page for each utility.

12.3.1 Using the btcreate Utility

To build a bootable SAS kernel on UFS or AdvFS file systems only, you must use the `btcreate` utility. This section provides an overview of the information you must have to create the SAS kernel on tape.

The `btcreate` utility provides both a noninteractive and interactive user interface. Both require that you have superuser privileges before using. To execute, this utility requires 156000 blocks (512 bytes per block) of disk space in the `/usr` directory.

12.3.1.1 Gathering Information

To prepare for a `btcreate` session, you must have the following information available:

- Name of the configuration file in the `/usr/sys/conf` directory.
- Name of the disk partition (for example, `rz2e`) where the miniroot file system is to reside. Minimum size needed on the disk is 30720 blocks (512 bytes per block). This disk partition should not be mounted when `btcreate` is executed.
- Name of the tape device, for example `nrmt0h`, where the SAS kernel and file systems are to reside.

- Device name, mount point, and type of each file system (UFS or AdvFS) that you want to back up to the tape device. The following shows valid UFS and AdvFS entries:

  UFS:

  ```
  /dev/rz1a  /     ufs
  /dev/rz1g  /usr  ufs
  ```

  AdvFS:

  ```
  root_domain#root  /     advfs
  usr_domain#usr    /usr  advfs
  ```

  Note

  Do not select swap partitions for file system backups.

- An *addlist_file*, which lists the files or directories you want to include on the miniroot file system.
- An *fslist_file*, which specifies the file systems to back up.
- A `/usr/lib/sabt/sbin/custom_install.sh` script, if you want to customize the restored system image. The file must be written in the Bourne shell language (`sh1`) as it is the only shell provided on the miniroot file system. The `btcreate` utility copies the `custom_install.sh` file onto tape and places it in the `sbin` directory on the miniroot file system. The `btextract` utility invokes the `custom_install.sh` script before exiting.

12.3.1.2 Creating the SAS Kernel

To create the SAS kernel, the `btcreate` utility copies the `/usr/sys/conf/`*`YOUR_SYSTEM_NAME`* configuration file to `/usr/sys/conf/`*`YOUR_SYSTEM_NAME.BOOTABLE`* and modifies it as follows:

```
config    vmunix   root  on md
pseudo-device      memd  30720
```

These modifications indicate that a memory file system of 30720 is being configured. The memory file system and the disk partition where the miniroot file system reside are equivalent in size.

After modifying the configuration file, the `btcreate` utility executes the `doconfig` command and moves the bootable kernel to the `/usr/sys/bin`

directory. For information on syntax format and flags, see the btcreate reference page.

12.3.2 Using the btextract Utility

The btextract utility is a shell script that restores file systems from tapes that contain the bootable Standalone System (SAS) kernel. The SAS kernel is created using the btcreate utility. You have the option of performing a DEFAULT restore or an ADVANCED restore of the system.

Performing a DEFAULT restore, you can duplicate the customized system on more than one machine of the same hardware platform type; however you cannot specify which disk partitions to use for the restore operation. Instead, the btextract utility restores file systems using the disk partition information gathered during the btcreate session; all existing information is overwritten.

Performing an ADVANCED restore, you can specify which disk partition to use, but the customized system can only be duplicated on a machine of the same hardware platform type.

To use the btextract utility, place the system in a halt state, initialize the system, then boot from the tape as follows:

```
>>> init
>>> show dev
>>> boot -fl "nc" MKA500
```

In the previous example, the show dev command provides the device name under BOOTDEV and MKA500 is the BOOTDEV.

After the initial boot is complete, the shell invokes the btextract utility. If you created a /usr/lib/sabt/sbin/custom_install.sh script during the btcreate session, the btextract utility invokes the custom_install.sh script before exiting. See the btcreate reference page for more information.

After the btextract utility completes its task, you must shut down the system, then reboot the system from the restored disk as follows:

```
# shutdown -h now
>>> boot DKA100
```

In this example, DKA100 is the BOOTDEV.

For more information and examples, see the btextract reference page.

12.4 Backing Up Data

It is important that all the files on your system, data files as well as system files, be protected from loss. Therefore, you should back up your entire system, including the system software. Most system files are static; that is, once they are installed they do not often change. Therefore, they do not need to be backed up as frequently as data files, which are dynamic, meaning they change constantly. Incremental backups are also possible.

Each file system backup is a single process. To ease the backup process, organize your file systems so that dynamic files are on file systems that are backed up regularly and static files are on file systems that are backed up occasionally. You may find that you have dynamic files on file systems that are backed up occasionally. If this happens and you wish to back them up regularly, just prior to performing a backup, copy the frequently changing files to systems that are backed up regularly. This allows you to back up those files without backing up an entire file system. You could write a shell script to automate these tasks for you.

The dump command copies all designated UFS file systems, or individual files and directories changed after a specified date, to a file, pipe, magnetic tape, disk, or diskette. The vdump command copies all AdvFS filesets. Refer to Chapter 8 for information on copying AdvFS file systems. You must have superuser privileges to use the dump command.

Note

To produce valid backups on a UFS file system, you must back up a file system while it is inactive. It is recommended that you unmount the file system and check it for consistency. As an added precaution, put the system into single-user mode before starting your backup operations. This is not true for AdvFS. Refer to Chapter 8 for information on restoring AdvFS file systems.

The remainder of this section describes the procedure for shutting down a system and unmounting and checking the integrity of a file system.

You can back up the system while in either multiuser mode or single-user mode. However, backups performed on file systems actively being modified might corrupt the backup data. The dump command operates by checking the inodes of the files you want to back up. The inodes contain data such as table entries and other statistics. When you use the dump command to back up files in a file system, an inode is attached to each file. If the system or user activity changes a file after the inode data is recorded, but before the file is backed up, the backup may be corrupted.

To shut down the system, unmount a file system, and check the integrity of a file system:

1. Shut down the system.

   For example, to shut down the system in 5 minutes and give users periodic warning messages, enter:

   ```
   # /usr/sbin/shutdown +5 'System going down to perform backups'
   ```

2. Use the umount command with the –a option to unmount the file systems that you want to back up:

   ```
   # /sbin/umount -a
   ```

   Note that the root file system remains mounted.

3. Use the fsck command to ensure the integrity of the file system.

   For example, to check a file system for an RZ57, unit 0, partition c, enter:

   ```
   # /sbin/fsck -o /dev/rz0c
   ```

This chapter describes a backup strategy that uses the dump and restore group of backup commands. Other backup strategies are possible. For example, you could use the find command to produce a list of files that must be backed up and pipe the list to a backup program such as tar or cpio. Refer to the find(1), tar(1), and cpio(1) reference pages for more information.

12.4.1 Choosing a Backup Schedule

When deciding how often to back up each file system, you should think about the balance between the potential loss of user time and data and the time it takes you to perform backups. Ask yourself the question, "How much information can I afford to lose?" The answer will help you determine your minimum backup interval. On most systems the backup interval is daily, but you can choose any other interval.

It is not necessary to back up all the files in a file system at each backup. Back up only those files that have changed since a previous backup; this is called an incremental backup. Using the dump and restore commands, you can perform up to nine levels of incremental backups. For example, while a level 0 dump backs up an entire file system, a level 1 dump backs up only those files since the last level 0 dump, and a level 7 dump backs up only those files since the last lower level dump.

To integrate incremental backups into your file backup schedule, you need to balance the time and tape space required for backup against the amount of time it could take you to restore the system in the event of a system

failure. For example, you could schedule backup levels following the 10-day sequence:

[0 1 2 3 4 5 6 7 8 9]

On the first day you save an entire file system (level 0). On the second day you save changes since the first backup and so on until the eleventh day when you restart the sequence. This makes the amount of time spent and data saved on each backup relatively small each day except the first; however, if a system failure on the tenth day requires that you restore the entire system, you must restore all ten tapes.

Most systems follow some variant of the common Tower of Hanoi backup schedule. Once a month you make a level 0 dump to tape of all the regularly backed up file systems. Then once a week, you make a level 1 dump to start a daily sequence of:

[...3 2 5 4 7 6 9 8 9 9 ...]

If you do backups only once a day on the weekdays, you end up with a monthly backup schedule as follows:

[0 1 3 2 5 4 1 3 2 5 4 ...]

This schedule, although slightly complex, requires that you restore at most four tapes at any point in the month if a system failure corrupts files. Of course, doing a level 0 dump daily requires that you restore at most one tape at any point, but requires a large amount of time and tape storage for each backup. On most days in the Tower of Hanoi schedule, very little time and tape storage are required for a backup.

12.4.2 Performing a Full Backup

You should set up a schedule for performing a full backup of each file system on your entire system, including all the system software. A conservative schedule for full system backups is to do one with each normal level 0 dump (using Tower of Hanoi, once a month), but you can set any schedule you like within the reliability of your storage media, which is about two years for magnetic tapes. To back up your file system, use the dump command, which has the following command syntax:

dump *options filesystem*

The *options* parameter specifies a list of flags and their arguments and the *filesystem* parameter specifies the file system to be backed up. You should specify the file system with a full pathname. The dump command can back up only a single file system at a time, but there may be several dump processes simultaneously writing files to different tape devices.

The dump(8) reference page describes the command options that you use to specify the characteristics of the tape device, such as block size, tape

storage density, and tape length. The following list describes the most commonly used options to the dump command:

–*integer*	Specifies the dump level as an integer (0-9). A dump level of 0 causes a full dump of the specified file system. All other dump levels cause an incremental backup. That is, only files that have changed since the last dump of a lower dump level are backed up. The /etc/dumpdates file contains a record of when the dump command was used on each file system at each dump level. The –u option to the dump command updates the dumpdates file.
–f *dump_file*	Writes the dump to the device specified by *dump_file* instead of to the default device, /dev/rmt0h. When *dump_file* is specified as a dash (-), the dump command writes to the standard output.
–u	Updates the /etc/dumpdates file with the time of the dump and the dump level for the file system in the backup. You use this file during incremental dumps (by using the dump level option) to determine which files have changed since a particular dump level. You can edit the /etc/dumpdates file to change any record or fields, if necessary. The dump(8) reference page describes the format of this file.

To back up your entire file system to the default backup device, use the dump command for each file system on your machine. The dump command has the following command syntax:

dump -0u *filesystem*

The *filesystem* parameter specifies the name of a file system on your machine. The –0u option causes a level 0 dump and updates the /etc/dumpdates file with the time and date of the backup for each file system. This creates an initial point on which to base all future incremental backups until the next full or level 0 dump. Note that each file system must be backed up individually.

For example, if you want to perform a level 0 dump of the root, /usr, and /projects file system partitions, follow these steps:

1. To back up the root file system, load a tape into your tape drive and enter:

   # **dump -0u /**

   After completing the backup, remove the tape from your tape drive.

2. To back up the /usr file system, load a new tape into your tape drive and enter:

   # **dump -0u /usr**

   After completing the backup, remove the tape from your tape drive.

3. To back up the /projects file system, load a new tape into your tape drive and enter:

   # **dump -0u /projects**

You can either back up each file system on an individual tape, or you can back up multiple file systems on one tape by specifying the no-rewind device, /dev/nrmt0h, as the output device. The following examples show the root, /usr, and /projects file systems being backed up on one tape:

```
# dump -0uf /dev/nrmt0h /
# dump -0uf /dev/nrmt0h /usr
# dump -0uf /dev/nrmt0h /projects
```

The previous example may require additional media management to cross-reference dump files with tapes, especially when a single dump file spans media. Exercise care when labeling this type of backup media.

12.4.3 Performing an Incremental Backup

You should set up a routine as part of your backup schedule to make it easier to remember which backup to do each day. This routine should include a mechanism for logging your backups and their dump level and for listing the tapes on which they are made. Because of the chance of system corruption, you should not keep this information on the computer system.

Once you have established a system for making incremental backups, the procedure is simple. Assume you use the following backup schedule to do a daily backup of /usr:

```
0 1 9 9 9 1 9 9 9 …
```

On Monday, perform a level 0 dump:

dump -0u /usr

On Tuesday, perform a level 1 dump:

```
# dump -1u /usr
```

The level 1 dump backs up all the files that changed since Monday. On Wednesday through Friday you perform a level 9 dump (which always backs up all the files that have changed since Tuesday's level 1 dump):

```
# dump -9u /usr
```

To perform the same level 9 dump to the tape device named `/dev/rmt1h` instead of the default tape device, use the –`f` option as shown in the following example:

```
# dump -9uf /dev/rmt1h /usr
```

The argument to the –`f` option specifies a tape device local to the system from which you are performing the dumps.

12.4.4 Performing a Remote Backup

Some machines in a networked system environment might lack a local tape drive that you can use for making backup tapes. You can use the `rdump` command to make backups on a remotely located tape device. The `rdump` command is identical to the `dump` command except that it requires the –`f` option to specify the machine name and an attached backup device. The `rdump` command has the following command syntax:

rdump -f *machine*:*device options filesystem*

The *machine* parameter specifies the name of the remote machine that has the backup device and *device* specifies the name of the backup device on that remote machine. The colon (`:`) between the *machine* and *device* parameters is necessary just as in other network file-addressing mechanisms.

The *options* parameter refers to the same list of flags available with the `dump` command.

The *filesystem* parameter refers to the local file system to be backed up.

The `rdump` command updates the `/etc/dumpdates` file on the local machine in the same way as does the `dump` command. The `rdump` command starts a remote server, `/usr/sbin/rmt`, on the remote machine to access the storage medium. This server process should be transparent. Refer to the `rmt`(8) reference page for more information.

To back up the `/projects` file system from `machine1` onto a tape drive on `machine2` with the attached backup device `/dev/rmt0h`, enter the following command from `machine1`. The name of `machine1` must be in the `/.rhosts` file of `machine2` to allow access from `machine1` to `machine2`.

```
# rdump -0uf machine2:/dev/rmt0h /projects
```

The dump(8) reference page describes the options to the rdump command.

12.4.5 Using Backup Scripts

You can automate the backup process by using shell scripts. The cron daemon can execute these shell scripts late in the evening when there is less chance of the dump commands making errors due to a changing system.

Backup shell scripts often perform the following tasks:

- Determine the dump level
- Warn the system of the dump
- Make a listing of tape contents
- Notify the operator upon completion

Some time during the day, load a tape into the tape drive. At the specified time, the cron daemon runs the backup shell scripts. When the shell procedures are finished, remove the backup tape and archive it.

Note that backup shell scripts are best used when the dump is small enough to fit on a single tape. You will need to specify the no-rewind device and the –N option to the dump command to inhibit the tape from automatically going off line when each dump completes. When dump reaches the end of the tape, it will take the tape off line and someone will need to be available to replace the tape.

12.5 Restoring Data

Occasionally, you will have to retrieve files from your backup tapes, and you will likely need to restore entire file systems at some time. If you have set up a good backup procedure, then restoring files or full file systems should be a simple task.

If a serious problem occurs, you may have to restore your entire system. Before restoring, determine what caused the problem with the system.

After determining the cause of the problem, reinstall your system from the initial boot tapes. The installation instructions that came with your system explain this procedure.

Once your system is up and running, restore the system to the state it was in just prior to the system crash. If you are using AdvFS, use the vrestore command. Refer to Chapter 8 for information on restoring the AdvFS file system. If you are using UFS, use the restore command to restore data from tapes made with the dump command. Because the dump

command saves a single file system at a time, you must execute the `restore` command for each file system you wish to restore. The `restore` command has the following command syntax:

restore *options*

The *options* parameter indicates the flags and arguments that you use to specify the characteristics of the tape device and special restore options. Refer to the `restore`(8) reference page for more information about these options. The following list describes the most commonly used options to the `restore` command:

`–i`
: The `i` (interactive) flag starts interactive restoration of files from the tape. After reading directory information from the tape, this option provides a shell-like interface that allows you to select the files you want to restore. The commands available in interactive mode are described in Section 12.5.3.

`–r`
: The `r` (restore) flag restores the entire contents of the file system on the backup tape into the current working directory. You should not do this except to restore an entire file system into an empty directory or to restore file system incremental dumps.

`–s`
: The `s` (skip) flag identifies which dump file on the media the `restore` command will use. This is useful when the dump media contains more than one dump image and not all of them will be restored. To effectively use this option, you must be consistent in the order in which you dump images to the tape. For example, if you dump multiple file systems to a single backup tape nightly, dump the file systems in the same order each night. This will assist you in locating a particular file or file system at restore time.

`–t` *names*
: The `t` (table of contents) flag creates a list of files and directories on the tape that matches the *names* argument. If you specify *names*, the `restore` command returns a list of the files and directories that are on the tape that matches the specified names. The *names* argument should be specified as `./`*filename* . For example, if the `.rhosts` file and the `staff` directory exist on the tape, the `restore`

`-t ./.rhosts ./staff` command will list the file and the directory. If you do not specify *names*, the `restore` command returns a complete listing of the backed up files on the tape.

–x *names*

The `x` (extract) flag restores from the tape the files and directories specified by the *names* argument. The *names* argument contains a list of files and directories to be restored from the tape. Specify `names` as `./`*filename* . For example, the `restore -x ./.rhosts ./staff` command will restore the `.rhosts` file and the `./staff` directory. If *names* specifies a directory name, then all the files in the directory are recursively restored.

–f *dump_file*

The `f` flag used with the *dump_file* argument restores the dump from the device specified by the *dump_file* argument instead of the default device, `/dev/rmt0h`.

–F *command_file*

The `F` flag used with the *command_file* argument specifies a file from which interactive restore commands are read. You should use this option in conjunction with the –`i` option.

If you are restoring a file system other than root or `/usr`, go to Section 12.5.1. If you are restoring the root and `/usr` file systems, go to Section 12.5.5. If the `/var` directory is on a separate file system than `/usr`, go to Section 12.5.5.

12.5.1 Restoring a File System

There may be times when you will need to restore a file system. This section describes a general procedure for restoring a file system. To restore individual files, go to Section 12.5.2.

When you restore a UFS file system, you create a new file system and restore the files from the dump files by using the following command syntax. Refer to Chapter 8 for information on restoring an AdvFS file system.

newfs *raw_device*

mount *block_device* [*filesystem*]

cd *filesystem*

restore -Yrf *dump_file*

If the disk does not have a label, write the label by using the `disklabel` command before you create the new file system. Use the following command syntax to determine if the disk has a label:

disklabel -r *disk*

Writing a label with customized partition table settings may affect the entire disk. Use the following command syntax to write the default disk partition table:

disklabel -rw *disk disk_type*

The *disk* parameter specifies the disk that includes the device mnemonic and unit number. The *disk_type* parameter specifies the type of disk associated with the *disk* as described in the `/etc/disktab` file.

Invoke the editing option of the `disklabel` command to use the customized partition table settings. Refer to Chapter 7 or to `disklabel`(8) for more information.

The *raw_device* parameter specifies the full raw device pathname of the disk device on your system. The *block_device* parameter specifies the full block device pathname of the disk device on your system. The *filesystem* parameter specifies the full pathname of the file system you want to make available. The *dump_file* parameter specifies the full pathname of the file containing the dump data.

The following example shows the commands you use to restore a file system called `/usr/projects` on an RZ57 disk from a tape:

```
# disklabel -rw rz1 rz57
# newfs /dev/rrz1c
# mount /dev/rz1c /usr/projects
# cd /usr/projects
# restore -Yrf /dev/rmt0h
```

12.5.2 Restoring Files

When users lose files, they ask their system administrator to restore those files. Users may also ask you to restore an earlier version of a file. Whatever the reason for a file restoration, determine which tape contains the correct version of the file. If you are restoring a file on UFS, use the `restore` command to restore the file. If you are restoring a file on AdvFS, refer to the `vrestore`(8) reference page for information.

By asking when the file was lost and when it was last modified, you can use your backup log to determine which tape contains the most recent version of the wanted file. Use the –t option with the restore command to determine whether a file is on the selected tape. Use the following syntax:

restore -t *./filename*

The –t option creates a list of files and directories on the tape that matches the *./filename* argument. For example, to list the contents of the working subdirectory of the /usr file system on a particular backup tape, load the tape and enter:

```
# restore -t ./working
```

To create a list of the entire contents of a backup tape, load the backup tape and enter:

```
# restore -t
```

Make a listing of each backup tape after you create it. This verifies a successful backup and gives you a place to look up what files are on the tape.

After determining the location of the file, create a new directory for the file. If you restore the file into an existing directory and the file already exists, the restored file will overwrite the existing file. Restore the file by using the following form of the restore command:

restore -x *./filename*

The file will be restored into your current working directory.

For example, to restore the working/old.file file from a /usr file system backup tape into your current directory, load the backup tape and enter:

```
# restore -x ./working/old.file
```

To restore the entire contents of the working subdirectory from the same tape, enter:

```
# restore -x ./working
```

If your dump media contains multiple dump images, you need to know the sequence of the dump images in order to restore a file from one of the images. To examine the contents of the first dump image on the media, load the tape and enter:

```
# restore -ts 1
```

The –s option followed by the number 1 specifies the first dump image.

For example, to restore the working/old.file file from a /usr file system, which is the third dump image on the backup tape into your current directory, load the backup tape and enter:

```
# restore -xs 3 ./working/old.file
```

12.5.3 Restoring Files Interactively

To ease the task of restoring multiple files, use the –`i` option to the `restore` command. This option starts an interactive `restore` session. The interactive mode has commands similar to shell commands.

To begin an interactive `restore` session, enter:

```
# restore -i
```

The system responds with the following prompt:

```
restore >
```

The following command-line options are available in the interactive restore mode:

`ls [` *`directory`* `]`	Lists files in the current or specified directory. Directory entries end with a / (slash). Entries that have been marked for reading begin with an * (asterisk).
`cd [` *`directory`* `]`	Changes the current directory to the directory specified by *`directory`*.
`pwd`	Lists the pathname of the current directory.
`add [` *`files`* `]`	Adds the files in the current directory or the files specified by *`files`* to the list of files recovered from the tape. Once they are specified to be read by the `add` command, files are marked with an * (asterisk) when they are listed with the `ls` command.
`delete [` *`files`* `]`	Deletes all the files in the current directory or the files specified by *`files`* from the list of files recovered from the tape.
`extract`	Restores from the tape the files that are marked to be read into the current working directory. The `extract` command prompts you for the logical volume that you want to mount (usually 1), and whether the access modes of the dot (.) are affected;

	answer `yes` when you are restoring the entire `root` directory.
`setmodes`	Sets owner, access modes, and file creation times for all directories that have been added to the files-to-read list; no files are recovered from the tape. Use this command to clean up files after a `restore` command has been prematurely aborted.
`verbose`	Toggles verbose mode. In verbose mode, each file name is printed to the standard output. By default, verbose mode is set to off. This is the same as the –`v` command line option to the `restore` command.
`help`	Lists a summary of the interactive commands.
`?`	Lists a summary of the interactive commands.
`what`	Lists the tape header information.
`quit`	Quits the interactive restore session.
`xit`	Exits from the interactive restore session. The `xit` command is the same as the `quit` command.

To interactively restore the `./working/file1` and `./working/file2` files from a backup tape, load the tape and enter:

```
# restore -i
```

Once in interactive mode, follow these steps to add the files to the list of files to be extracted:

1. Change to the `working` directory:

   ```
   restore > cd working
   ```

2. At the prompt, enter the file name.

   ```
   restore > add file1
   ```

3. Enter the name of the second file.

   ```
   restore > add file2
   ```

4. Extract the files.

   ```
   restore > extract
   ```

5. You are prompted for the logical volume you want to mount; usually you respond to this prompt with 1.

```
You have not read any tapes yet.
Unless you know which volume your file(s) are on you can start
with the last volume and work towards the first.

Specify next volume #: 1
```

   You are then asked whether the extract affects the access modes of the dot (.). For this example, reply with n.

```
set owner/mode for '.'? [yn] n
```

6. Once the files are extracted, quit the interactive session.

```
restore > quit
```

The file1 and file2 files are now in the current directory.

You can automate this procedure in a command file that is read by the –F option to the restore command. For example, the following command file, named restore_file, performs the restore operation shown in the previous example:

```
cd working
add file1
add file2
extract
1
n
quit
```

To read and execute this shell script, enter the following command:

```
# restore -iF restore_file
```

The result of the procedure in this script is identical to that of the previous interactive restore session.

12.5.4 Performing Remote Restores

There may be times when you need to perform remote restores. You can use the rrestore command to perform restores to local directories from a remote tape device. The rrestore command requires the –f option to specify the machine name and its backup device. The restore command has the following syntax:

rrestore -f *machine*:*device* [*options*]

The *machine* argument specifies the name of the remote machine where the backup device is attached, and *device* specifies the name of the backup device on that remote machine. The colon (:) between *machine* and *device* is necessary just as in other network file-addressing mechanisms.

The *options* for the rrestore command are the same as for the restore command. See Section 12.5 for a description of the options.

To restore the ./working/file1 file onto the local directory on machine1 from a backup tape mounted on machine2 where the backup device /dev/rmt0h is attached, enter the following command from machine1. The name machine1 must be in the /.rhosts file of machine2 to allow access from machine1 to machine2.

```
# rrestore -xf machine2:/dev/rmt0h ./working/file1
```

The rrestore command starts a remote server, /usr/sbin/rmt, on the remote machine to access the storage medium. This process should be transparent. Refer to the rmt(8) reference page for more information. See Section 12.5 for a description of the options to the rrestore command.

12.5.5 Restoring the root and /usr File Systems

This section describes a procedure for restoring the root and /usr file systems. The root file system must be restored before you can restore the /usr file system. If the /var directory is on a file system other than /usr, repeat the steps in this section for restoring /var.

The procedure in this section requires that you have access to the most recent dump files of your root and /usr file systems. You should use this procedure only when a catastrophic error occurs on the system disk, such as a disk crash or when an inadvertent deletion of either the root or /usr file systems renders the system inoperative.

The following example assumes that you are restoring from level 0 dump files and that you are using the text-based (or character cell) interface to the task.

1. Load the installation software. For removable media such as tape or CD–ROM, insert the media into the appropriate drive. For RIS installations, verify that the inoperative system has been registered on the RIS server. See *Sharing Software on a Local Area Network* for details. If the dump file is located on a remote system, include the hostname of the inoperative system in the /.rhosts file of the remote system. For security reasons, be sure to delete the hostname from the /.rhosts file after the restore operation has completed.
2. Boot the Digital UNIX software as described for your processor and distribution media in the *Installation Guide*. If your system had a graphical interface, the Installation Setup screen would be displayed, rather than the following menu. However, in both cases you would select the UNIX Shell option.
3. Select the UNIX Shell option at the prompt.

4. Create the special files for the root file system device and dump file device.

   - If you are restoring dump files from a local system, change to the `/dev` directory and use the `MAKEDEV` command with the following command syntax:

     MAKEDEV *mnemonic*

     The *mnemonic* parameter refers to a device mnemonic. See Appendix A for a list of the supported device mnemonics. For example, to create the special files for an RZ57 disk, unit number 0, and a TLZ06 tape, unit number 5, enter:

     ```
     # cd /dev
     # ./MAKEDEV rz0 tz5
     ```

   - If you are restoring dump files from a remote system, change to the `/dev` directory and use the `MAKEDEV` command with the following command syntax:

     MAKEDEV *mnemonic*

     The *mnemonic* parameter refers to a device mnemonic. See Appendix A for a list of the supported device mnemonics. For example, to create the special files for an RZ57 disk, unit number 0, enter:

     ```
     # cd /dev
     # ./MAKEDEV rz0
     ```

   After creating the system disk special file, configure the network by configuring the network interface and creating the hostname database (`/etc/hosts`). Use the `ifconfig` command with the following syntax to configure the network interface:

   ifconfig *interface_id local_address mask*

   The *interface_id* parameter refers to the network device mnemonic. Refer to the `uerf`(8) reference page for information about obtaining an interface ID. The *local_address* parameter specifies the Internet address for the local host. The `netmask` *mask* parameter specifies how much of the address to reserve for subdividing networks into subnetworks. You can get the `netmask` value by entering the `ifconfig` command on a system within the immediate area. For example, to get the `netmask` value from the system `ln0`, enter:

   ```
   # ifconfig ln0
   ```

   Refer to the `hosts`(4) and `ifconfig`(8) reference pages for more information. Enter the following commands to configure the network for the system `localsystem`, with an Internet address of 120.105.5.1,

connected by an Ethernet interface to the remote system `remotesystem`, with an Internet address of 120.105.5.2:

```
# cd /etc
# echo "127.0.0.1 localhost" >> hosts
# echo "120.105.5.2 remotesystem" >> hosts
# ifconfig ln0 120.105.5.1 netmask 0xfffffc00
```

Some older systems broadcast all 0s instead of all 1s. In this situation, you must also specify the `broadcast address`.

5. Change to the root directory.

   ```
   # cd /
   ```

6. If the disk does not have a label, which could occur if the disk was physically damaged or replaced, write the default disk partition tables and bootstrap programs. The disk partitions and bootstrap programs should be operational. To determine if the disk has a valid label, use the `disklabel` command with the following syntax:

   disklabel -r *disk*

   Use the `disklabel` command with the following syntax to write the default disk partition table:

   disklabel -rw *disk disk_type*

   The *disk* parameter specifies the disk that includes the device mnemonic and unit number. The *disk_type* parameter specifies the type of disk associated with *disk* as described in the `/etc/disktab` file. For example, to write the default disk partition tables on an RZ57 disk, unit 0, enter the following command:

   ```
   # disklabel -rw rz0 rz57
   ```

   Note

   The `disklabel` command used in this procedure writes the default disk partition tables to the disk. Writing a label with customized partition table settings may affect the entire disk. If the disk you are restoring has customized partition table settings, invoke the editing option of the `disklabel` command. Refer to Chapter 7 or to the `disklabel`(8) reference page for more information.

7. Create a new root file system by using the following command syntax:

   newfs *raw_device*

   The *raw_device* parameter specifies the full raw device pathname of the disk device on your system. For example, to create a new file system on an RZ57, unit 0, enter:

```
# newfs /dev/rrz0a
```

8. Mount the file system by using the following command syntax:

   mount *block_device* [/mnt]

   The *block_device* parameter specifies the full block device pathname of the disk device. For example, to mount the file system created in the previous step, enter:

   ```
   # mount /dev/rz0a /mnt
   ```

9. Restore the file system:

   - If you are restoring dump files from a local file system, change to the /mnt directory, insert the medium containing the dump file, and enter the restore command with the following command syntax:

     restore [-Yrf] [*dumpfile*]

     The *dumpfile* parameter specifies the pathname of the file that contains the dump data. For a tape, you would enter the following commands:

     ```
     # cd /mnt
     # restore -Yrf /dev/rmt0h
     ```

   - If you are restoring dump files from a remote system, change to the /mnt directory and use the rsh command with the following syntax:

     rsh [*remote_hostname*] ["dd if= *dumpfile* bs= *blocksize*" | restore -Yrf -]

     The *remote_hostname* parameter specifies the host name of the remote system that contains the dump file. The *dumpfile* parameter specifies the full pathname of the dump file on the remote system, and the *blocksize* parameter is necessary for reading from a tape.

     The dump file must be read with the same block size as was used when writing to the tape. The default dump record size is 10 KB.

     For example, to restore a dump file on a TLZ06 from the remote system remotesystem that was written using the default block size, enter:

     ```
     # cd /mnt
     # rsh remotesystem  "dd if=/dev/rmt0h bs=10k"  | restore -Yrf -
     ```

10. Change to the root directory and unmount the file system.

    ```
    # cd /
    # umount /mnt
    ```

11. Restore the /usr file system.

    - If the /usr file system is on the same device as root, the process is similar to steps 7 through 10. To restore the /usr file system on the g partition of the same device as the root file system from the same tape device, enter the following sequence of commands. If you are using AdvFS, this step will not work. Use the procedure in step 11a.

      ```
      # newfs /dev/rrz0g
      # mount /dev/rz0g /mnt
      # cd /mnt
      # restore -Yrf /dev/rmt0h
      # cd /
      # umount /mnt
      ```

      a. Use the following procedure to restore the /usr directory on AdvFS from a tape mounted on rmt0 to a drive other than root:

         ```
         # cd /dev
         # MAKEDEV rz1
         # cd /
         # disklabel -rw rz1 rz57
         # mkfdmn /dev/rz1c usr_domain
         # mkfset usr_domain usr
         # mount -t advfs usr_domain#usr /usr
         # vrestore -x -D /usr
         ```

    - If the /usr file system is on a different device from root, the process is similar to steps 4 through 10. To restore /usr on an RZ57, unit 1, c partition from the same tape device, enter the following sequence of commands:

      ```
      # cd /dev
      # MAKEDEV rz1
      # cd /
      # disklabel -rw rz1 rz57
      # newfs /dev/rrz1c
      # mount /dev/rz1c /mnt
      # cd /mnt
      # restore -Yrf /dev/rmt0h
      # cd /
      # umount /mnt
      ```

12. Halt the system.

    ```
    # halt
    ```

13. Boot the system as described for your processor and distribution media in the *Installation Guide*.

12.5.5.1 Local Restoration Example

The following text-based example shows a portion of the restoration procedure for the root and /usr file systems to an RZ57, unit 0, from a TLZ06, unit 5. The backslashes in this example indicate line continuation and are not in the actual display.

```
.
.
.
Select one of the following options:

      1) Default Installation
      2) Custom Installation
      3) UNIX Shell

Enter your choice: 3

# cd /dev
# MAKEDEV rz0 tz5
MAKEDEV: special file(s) for rz0:
rz0a rrz0a rz0b rrz0b rz0c rrz0c rz0d rrz0d rz0e rrz0e rz0f \
rrz0f rz0g
rrz0g rz0h rrz0h
MAKEDEV: special file(s) for tz5:
rmt0l
nrmt0l
rmt0h
nrmt0h
rmt0m
nrmt0m
rmt0a
nrmt0a

# cd /
# disklabel -rw rz0 rz57
# newfs /dev/rrz0a
Warning: 575 sector(s) in last cylinder unallocated
/dev/rrz0a:     40960 sectors in 39 cylinders of 15 tracks, \
71 sectors
21.0MB in 3 cyl groups (16 c/g, 8.72MB/g, 2048 i/g)
super-block backups (for fsck -b #) at:
 32, 17152, 34272,
# mount /dev/rz0a /mnt
# cd /mnt
# restore -Yrf /dev/rmt0h
# cd /
# umount /mnt
# newfs /dev/rrz0g
Warning: 105 sector(s) in last cylinder unallocated
/dev/rrz0g:   614400 sectors in 577 cylinders of 15 tracks, 71 \
```

```
sectors
        314.6MB in 37 cyl groups (16 c/g, 8.72MB/g, 2048 i/g)
super-block backups (for fsck -b #) at:
 32, 17152, 34272, 51392, 68512, 85632, 102752, 119872, 136992,
 154112, 171232, 188352, 205472, 222592, 239712, 256832, \
 272672, 289792,
 306912, 324032, 341152, 358272, 375392, 392512, 409632, \
 426752, 443872,
 460992, 478112, 495232, 512352, 529472, 545312, 562432, \
 579552, 596672,
 613792,
# mount /dev/rz0g /mnt
# cd /mnt
# restore -Yrf /dev/rmt0h
# cd /
# umount /mnt
# halt
syncing disks...  done
halting....  (transferring to monitor)
```

12.5.5.2 Remote Restoration Example

The following text-based example shows a portion of the restoration procedure for the root and /usr file systems to an RZ57, unit 0, from a remote tape device. The remote system is called remotesystem and has an Internet address of 120.105.5.2. The local system is called localsystem and has an Internet address of 120.105.5.1.

```
.
.
Select one of the following options:

      1) Default Installation
      2) Custom Installation
      3) UNIX Shell

Enter your choice: 3

# cd /dev
# MAKEDEV rz0
MAKEDEV: special file(s) for rz0:
rz0a rrz0a rz0b rrz0b rz0c rrz0c rz0d rrz0d rz0e rrz0e rz0f rrz0f rz0g \
rrz0g rz0h rrz0h
# cd /etc
# echo "127.0.0.1 localhost" >> hosts
# echo "120.105.5.2 remotesystem" >> hosts
# ifconfig ln0 120.105.5.1 netmask 0xfffffc00
# cd /
# disklabel -rw rz0 rz57
# newfs /dev/rrz0a
Warning: 575 sector(s) in last cylinder unallocated
/dev/rrz0a:     40960 sectors in 39 cylinders of 15 tracks, 71 sectors
        21.0MB in 3 cyl groups (16 c/g, 8.72MB/g, 2048 i/g)
super-block backups (for fsck -b #) at:
 32, 17152, 34272,
# mount /dev/rz0a /mnt
```

```
# cd /mnt
# rsh remotesystem "dd if=/dev/rmt0h bs=10k" | restore -Yrf -
1743+0 records in
1743+0 records out
# cd /
# umount /mnt
# newfs /dev/rrz0g
Warning: 105 sector(s) in last cylinder unallocated
/dev/rrz0g:     614400 sectors in 577 cylinders of 15 tracks, 71 sectors
        314.6MB in 37 cyl groups (16 c/g, 8.72MB/g, 2048 i/g)
super-block backups (for fsck -b #) at:
 32, 17152, 34272, 51392, 68512, 85632, 102752, 119872, 136992,
 154112, 171232, 188352, 205472, 222592, 239712, 256832, 272672, 289792,
 306912, 324032, 341152, 358272, 375392, 392512, 409632, 426752, 443872,
 460992, 478112, 495232, 512352, 529472, 545312, 562432, 579552, 596672,
 613792,
# mount /dev/rz0g /mnt
# cd /mnt
# rsh remotesystem "dd if=/dev/rmt0h bs=10k" | restore -Yrf -
19922+0 records in
19922+0 records out
# cd /
# umount /mnt
# halt
syncing disks...  done
halting....  (transferring to monitor)
```

13

Administering the System Accounting Services

This chapter describes how to set up and use the system accounting services. The accounting services are shell scripts and commands you use to manipulate an accounting database to obtain a diagnostic history of system resource use and user activity and to create report files.

By using the accounting services, you can obtain accounting information for the following:

- Amount of connect time
- Amount of CPU time
- Number of processes spawned
- Number of connect sessions
- Amount of memory usage
- Number of I/O operations and number of characters transferred
- Disk space usage (in blocks)
- Amount of modem usage and telephone connect time
- Printer usage, including the number of printing operations and amount of printed matter, according to user name or printer name

You can set up accounting so that information is collected automatically on a periodic basis. You can also manually invoke accounting shell scripts and commands to obtain accounting information when you need it.

13.1 Accounting Overview

If accounting is enabled, the kernel and other system processes write records to the accounting database files, which are the source of all the accounting information.

The accounting database files are located in the `/var/adm` directory and include the following files:

File	Description
wtmp	The login/logout history file
utmp	The active connect session file
pacct	The active process accounting file
dtmp	The disk usage file

The accounting scripts and commands access the records in the accounting database files and reformat them so that you can use the records for purposes such as archiving, diagnostic analysis, or resource billing.

The various accounting shell scripts and commands also can do the following:

- Format the database file records
- Create new source files from the database file records
- Display the database file records
- Merge data from several files into a single formatted file
- Summarize data in files that you can use to create reports

You can redirect or pipe script and command output to files or to other scripts and commands.

System accounting allows you to distinguish between prime time and nonprime time. The system is used most during prime time and least during nonprime time. System use during nonprime time can be assessed at a lower rate than system use during prime time. You specify the period of nonprime time in the `/usr/sbin/acct/holidays` database file. Usually, if enabled, automatic accounting is performed during nonprime time.

The accounting period begins when the `/var/adm/pacct` file is created by the `startup` shell script when accounting is turned on or by the `runacct` script, which is usually run every day.

In command output, the order of date and time information is site dependent. You can change the order of date and time specifications by setting the NLTIME environment variable.

13.1.1 Accounting Shell Scripts and Commands

There are 14 accounting shell scripts and 20 accounting commands. The shell scripts often call the accounting commands or other shell scripts. The accounting commands and shell scripts create and write records to the accounting database files. Table 13–1 describes the accounting commands and shell scripts.

Table 13–1: Accounting Commands and Shell Scripts

Name	Type	Description
ac	Command	Displays connect session records.
acctcms	Command	Formats the binary command usage summary files.
acctcom	Command	Displays process accounting record summaries from the default pacct database file or a specified file.
acctcon1	Command	Summarizes the records in the wtmp file in ASCII format.
acctcon2	Command	Summarizes the contents of the files formatted by the acctcon1 command.
acctdisk	Command	Performs comprehensive disk usage accounting.
acctdusg	Command	Performs disk block usage accounting.
acctmerg	Command	Merges accounting record files.
accton	Command	Turns on process accounting.
acctprc1	Command	Displays records of acct type structure by user identification number and login name.
acctprc2	Command	Displays records of acct type structure by user identification number and full name.
acctwtmp	Command	Writes records to the /var/adm/wtmp file.
chargefee	Script	Writes a charge-fee record to the /var/adm/fee database file.
ckpacct	Script	Checks the size of the /var/adm/pacct active binary process accounting file to ensure that it is not too large.
diskusg	Command	Performs disk accounting according to user identification number.
dodisk	Script	Writes daily disk usage accounting records to the /var/adm/nite/dacct disk usage accounting database file.
fwtmp	Command	Displays the /var/adm/wtmp binary file records in ASCII format, allowing you to fix errors.
last	Command	Displays login information.
lastcomm	Command	Displays information about commands that were executed.
lastlogin	Script	Writes the date of the last login for all users to the /var/adm/acct/sum/loginlog file.
monacct	Script	Creates monthly summary accounting report files.

Table 13–1: Accounting Commands and Shell Scripts (cont.)

Name	Type	Description
nulladm	Script	Creates files that are owned by the adm user and group and that have 664 permission.
pac	Command	Displays printer accounting records.
prctmp	Script	Displays the /var/adm/acct/nite/ctmp connect session record file.
prdaily	Script	Collects and displays daily accounting records from various files.
printpw	Command	Displays the contents of the /etc/passwd file.
prtacct	Script	Formats in ASCII and displays a tacct daily accounting file.
remove	Script	Removes any /var/adm/acct/sum/wtmp*, /var/adm/acct/sum/pacct*, and /var/adm/acct/nite/lock* files.
runacct	Script	Invokes the daily accounting processes. This command periodically calls various accounting commands and shell scripts to write information to various accounting files.
sa	Command	Displays a summary of accounting records.
shutacct	Script	Turns off accounting.
startup	Script	Enables accounting processes.
turnacct	Script	Controls the creation of process accounting files.
wtmpfix	Command	Corrects date and time stamp inconsistencies in the /var/adm/wtmp file.

13.1.2 Accounting Files

Many binary and ASCII files are created and maintained by the kernel or by the accounting commands and shell scripts.

You should ensure that the accounting files, particularly those in binary format, do not become too large. Some extraneous files are produced by the accounting commands and shell scripts, but in general these files are temporary and exist only while the process is running. Under some circumstances (if a process terminates prematurely, for example), one or more temporary files can appear in one of the /var/adm subdirectories. You should check these subdirectories periodically and remove the unnecessary files.

Accounting files can become corrupted or lost. The files that are used to produce daily or monthly reports, such as the /var/adm/wtmp and /var/adm/acct/sum/tacct accounting database files, must have complete integrity. If these files are corrupted or lost, you can recover them from backups. In addition, you can use the fwtmp or the wtmpfix command to correct the /var/adm/wtmp file. Refer to Section 13.4.2 and Section 13.4.1 for more information. You can use the acctmerg command to fix errors in the /var/adm/acct/sum/tacct file. Refer to Section 13.9.2 for more information.

The /var/adm/acct/nite directory contains files that are reused daily by the runacct script. Some of these files have binary counterparts in the /var/adm/acct/sum directory, which contains the cumulative summary files that are updated by the runacct shell script and used by the monacct shell script to produce monthly reports.

Table 13–2 to Table 13–5 list the accounting files. The Name column specifies the file name and the table title specifies the directory pathname for the files. The Type column tells you if the file is an ASCII file or a binary file. The Description column provides a description of the file.

Table 13–2: Database Files in the /var/adm Directory

Name	Type	Description
dtmp	ASCII	Contains temporary output produced by the dodisk shell script.
fee	ASCII	Contains output from the chargefee shell script.
pacct	Binary	Specifies the active process accounting database file. If a process is called by a user, another process, or a script file, process information is written to this file.
pacct*n*	Binary	Specifies the alternate pacct file created by the turnacct switch command. The pacct database file becomes large quickly if a system has many users. A single pacct file is limited to 500 1024-block disk spaces. The size of these files is monitored by the runacct shell script. Each time a new pacct*n* file is created, the value *n* is incremented by one.
qacct	Binary	Contains queueing (printer) system accounting records. This file is used by the runacct shell script.
savacct	Binary	Specifies the file used by the sa command to store system process accounting summary records.

Table 13–2: Database Files in the /var/adm Directory (cont.)

Name	Type	Description
Spacct*n.mmdd*	Binary	Specifies the pacct*n* files produced by the runacct shell script for the month and day specified by *mm* and *dd*, respectively.
usracct	Binary	Specifies the file used by the sa command to store user process accounting summary records.
utmp	Binary	Specifies the active connect session accounting database file, which is written to if a user calls a process that produces a connect session.
wtmp	Binary	Specifies the cumulative login/logout accounting database file. If a user logs in to the system, connect time and user information is written to this file.

Table 13–3: Daily Files in the /var/adm/acct/nite Directory

Name	Type	Description
active	ASCII	Specifies the daily runacct shell script progress file. When the runacct shell script executes, information about its progress is written to this file. This file also contains error and warning messages.
active*mmdd*	ASCII	Specifies the daily runacct shell script error file for the month and day specified by *mm* and *dd*, respectively. This file is similar to the active file.
cklock	ASCII	Specifies the file the ckpacct shell script uses to ensure that more than one runacct shell script is not called during any 24-hour period. This file is removed each day if the runacct shell script has completed.
cms	ASCII	Specifies the active total daily command summary file. This file is the ASCII version of the /var/adm/acct/sum/cms file. This file is created by the acctcms command, which is called by the runacct shell script to rewrite the /var/adm/acct/sum/cms file records. The monacct shell script initializes this file.

Table 13–3: Daily Files in the /var/adm/acct/nite Directory (cont.)

Name	Type	Description
ctacct.*mmdd*	Binary	Specifies the connect accounting records in tacct.h format that are obtained from the connect session accounting records for the month and day specified by *mm* and *dd*, respectively. This file is temporary and is deleted after the daytacct file records are written for each accounting period.
ctmp	ASCII	Specifies the temporary login/logout record file. This file contains the output of the acctcon1 accounting command, which is called by the runacct shell script to rewrite the wtmp file records.
daycms	ASCII	Specifies the daily command summary file. This file is the ASCII version of the /var/adm/acct/sum/daycms binary file. The runacct shell script calls the prdaily shell script, which invokes the acctcms command to create the file.
daytacct	Binary	Contains the total accounting records in tacct.h format for the previous day.
dacct	Binary	Contains the weekly total disk usage accounting records when the acctdisk command is called by the dodisk shell script.
lastdate	ASCII	Specifies the last day that the runacct shell script was executed.
lineuse	ASCII	Contains terminal (tty) line connect times. This file provides line use statistics for each terminal line used during the previous accounting period.
lock	ASCII	Specifies the file used to ensure that the cron daemon does not call the runacct shell script more than once during any 24-hour period. This file is removed each day when the runacct shell script has completed.
log	ASCII	Contains diagnostic output that is produced when the runacct script invokes the acctcon1 command.
owtmp	Binary	Specifies the daily wtmp file after a correction by the wtmpfix command.
ptacct*n.mmdd*	Binary	Specifies the additional daily pacct*n* files for the month and day specified by *mm* and *dd*, respectively. These files are created if the daily pacct process accounting file requires more than 500 disk blocks.

Table 13–3: Daily Files in the /var/adm/acct/nite Directory (cont.)

Name	Type	Description
`reboots`	ASCII	Contains a list of system reboots during the previous accounting period.
`statefile`	Binary	Specifies the final `runacct` shell script execution state.
`wtmp.`*`mmdd`*	Binary	Specifies the fixed daily login/logout accounting database file for the month and day specified by *mm* and *dd*, respectively. Connect session records of users who logged in to the system during the previous day are written to this file.
`wtmperror`	ASCII	Contains any error messages produced when a `wtmp` file is fixed during the execution of the `wtmpfix` command.
`wtmperror`*`mmdd`*	ASCII	Contains any error messages produced when the `runacct` shell script detects an error during execution of the `wtmpfix` command for the month and day specified by *mm* and *dd*, respectively.

Table 13–4: Summary Files in the /var/adm/acct/sum Directory

Name	Type	Description
`cms`	Binary	Specifies the active total command summary file. When the `runacct` shell script is executed, records are written to this file to obtain the total command summary file.
`cmsprev`	Binary	Specifies the previous day's `/var/adm/acct/sum/cms` file.
`daycms`	Binary	Specifies the previous day's command summary file. When the `runacct` shell script is executed, monthly command summary records for the previous day are written to this file.
`loginlog`	ASCII	Contains a list of the last monthly login date for each user name.
`rprt`*`mmdd`*	ASCII	Specifies the daily accounting report for the month and day specified by *mm* and *dd*, respectively.
`tacct`	Binary	Specifies the cumulative total accounting file. This file is the total daily accounting file for system use. It is updated on a daily basis by the `runacct` shell script.

Table 13–4: Summary Files in the /var/adm/acct/sum Directory (cont.)

Name	Type	Description
`tacct`*mmdd*	Binary	Specifies the total accounting file for the month and day specified by *mm* and *dd*, respectively.
`tacctprev`	Binary	Specifies the previous day's `tacct` file. This file is the `tacct` binary file for the previous accounting period.

Table 13–5: Monthly Files in the /var/adm/acct/fiscal Directory

Name	Type	Description
`cms`*mm*	Binary	Specifies the active command summary file for the month specified by *mm*.
`fiscrpt`*mm*	ASCII	Specifies the accounting report for the month specified by *mm*.
`tacct`*mm*	Binary	Specifies the cumulative total accounting file. This file is the total accounting file for system use. It is updated on a monthly basis by the `monacct` shell script.

13.2 Setting Up Accounting

In a system environment where many users compete for system resources, Digital UNIX system accounting allows you to track system use. You must decide the quantity and type of information that you want to track. You also must decide if you want to enable automatic accounting. To enable automatic accounting, you specify accounting commands and shell scripts in the files in the `/usr/spool/cron/crontabs` directory.

Note

You must install the System Accounting Utilities subset to use accounting.

To obtain accounting information for all the machines in a network, you should set up accounting on a single machine. Use the following procedure to enable system accounting. The sections that follow describe these steps in detail.

1. Enable accounting in the `/etc/rc.config` file.
2. Create the `/var/adm/qacct` and `/var/adm/pacct` files.

3. Edit the /usr/sbin/acct/holidays file to specify prime time, nonprime time, and holidays.
4. To enable automatic accounting, modify the files in the /usr/spool/cron/crontabs directory to invoke accounting shell scripts and commands.

Resource accounting is discussed separately from printer accounting because the print driver software uses different servers, daemons, and routines. Setting up printer accounting is described in Chapter 11.

13.2.1 Enabling Accounting in the rc.config File

To enable accounting, you must add the following line to the /etc/rc.config file:

```
ACCOUNTING="YES"
```

You can use the rcmgr command to set the variable, as follows:

```
# rcmgr  set  ACCOUNTING  YES
```

You can start accounting without rebooting your system by using the startup command. Refer to Section 13.3 for more information.

13.2.2 Creating the qacct and pacct Files

You must create the /var/adm/qacct queueing accounting file and the /var/adm/pacct process accounting database file. Use the nulladm command to create the files.

13.2.3 Editing the holidays File

The /usr/sbin/acct/holidays file uses 24-hour time to specify prime time and nonprime time. The file also specifies holidays, which are included in nonprime time. Only the days Monday through Friday are included in prime time. You can assess system use during nonprime time at a lower rate than during prime time. If you enable automatic accounting, you should specify that the commands be executed during nonprime time.

If the /usr/sbin/acct/holidays file does not exist, you must create it. If the file exists, you must edit it to reflect your accounting needs.

You can set the NHOLIDAYS environment variable to specify the maximum number of holidays that you can include in the holidays file.

13.2.4 Modifying the crontab Files

To enable automatic accounting, you must use the `crontab` command to modify the files in the `/usr/spool/cron/crontabs` directory. The files in the `/usr/spool/cron/crontabs` directory contain commands that the `cron` daemon runs at specified times under a specific authority. For example, the commands in the `/usr/spool/cron/crontabs/root` file are run under `root` authority, and the commands in the `/usr/spool/cron/crontabs/adm` file are run under `adm` authority.

You can include the following commands and shell scripts in the `/usr/spool/cron/crontabs/adm` file:

`ckpacct`	This shell script checks the size of the `/var/adm/pacct` process accounting database file and ensures that it does not become too large.
`runacct`	This shell script includes other accounting shell scripts and commands and creates daily and monthly accounting files. You can modify the `runacct` shell script to remove the commands for the accounting features that you do not want.
`monacct`	This shell script creates monthly summary accounting files. You can modify the `monacct` shell script to remove the commands for the accounting features that you do not want.
`ac`	This command displays connect-time records. You can direct the output to a file. You can also add this command to the `runacct` shell script.
`pac`	This command displays printer accounting records. You can direct the output to a file. To enable printer accounting, refer to Section 13.8.

You can include the `dodisk` shell script in the `/usr/spool/cron/crontabs/root` file. The `dodisk` shell script creates disk usage accounting records and should be run once during nonprime time each week.

Refer to Chapter 4 and to the `crontab`(1) reference page for more information on submitting commands with the `crontab` command.

The following example shows part of a /usr/spool/cron/crontabs/adm file that includes accounting commands and shell scripts:

```
0 2 * * 1-6 /usr/sbin/acct/runacct  > /usr/adm/acct/nite/fd2log&
5 * * * * /usr/sbin/acct/ckpacct&
0 4 1 * *  /usr/sbin/acct/monacct&
10 3 * * * /usr/sbin/ac -p > /var/adm/timelog&
40 2 * * * /usr/sbin/pac -s&
```

The following example shows part of a /usr/spool/cron/crontabs/root file that includes the dodisk shell script:

```
0 3 * * 4 /usr/sbin/acct/dodisk  > /var/adm/diskdiag&
```

13.3 Starting Up and Stopping Accounting

The startup and shutacct shell scripts enable and disable the various accounting processes. The scripts invoke the acctwtmp program, which adds a record to the /var/adm/wtmp file by using the system name as the login name.

The startup shell script initializes the accounting functions and has the following syntax:

/usr/sbin/acct/startup

Note

You must ensure that the /var/adm/pacct file, which is created by the startup script, is owned by group adm and user adm and has 664 protection. If it does not have the correct ownership, the accton command will not work, and the following message will be displayed:

```
accton: uid/gid not adm
```

The shutacct script turns process accounting off and ensures that the accounting functions are halted before the system shuts down. The shutacct shell script has the following syntax:

/usr/sbin/acct/shutacct [*Reason*]

If the shutacct shell script is invoked, the *'Reason'* message is written to the ut_line field in the /var/adm/wtmp file shutdown record. Then, the turnacct off shell script is invoked to tell the kernel that its active accounting functions should be disabled.

13.4 Connect Session Accounting

When a user logs in or logs out, the `login` and `init` commands write the user login and logout history to records in the `/var/adm/wtmp` binary database file. The `/var/adm/utmp` binary database file is the active connect session file. All hangups, terminations of the `login` command, and terminations of the login shell cause the system to write logout records, so the number of logouts is often more than the number of sessions.

Connect session commands can convert the `/var/adm/wtmp` file records to useful connect session accounting records. You can obtain connect session accounting only if the `/var/adm/wtmp` file exists.

The formatted records in the `/var/adm/wtmp` file provide the following information about each connect session:

- User login name (from the `/etc/passwd` file)
- Line identification number (from the `/etc/inittab` file)
- The device name (for example, console or `tty23`)
- Type of entry
- Process identification number
- Process termination status
- Process exit status
- Time entry was made
- Host machine name

You can use the following two shell scripts and seven commands to obtain or modify information about system connect sessions:

Command	Description
`ac`	This command displays connect session records for the entire system and for each user.
`acctcon1`	This command summarizes connect session records and displays those records in ASCII format, using one line for each connect session.
`acctcon2`	This command uses the output of the `acctcon1` command to produce an accounting record file of the total connect session in ASCII format.
`acctwtmp`	This command enables you to write records to the `wtmp` file by entering them from the keyboard.

Command	Description
fwtmp	This command displays records from files with the utmp.h file structure.
last	This command displays login information.
lastlogin	This shell script updates the /var/adm/acct/sum/loginlog file to show the last date that each user logged in.
prctmp	This shell script displays the contents of the session-record file (usually /var/adm/acct/nite/ctmp) that the acctcon1 command created.
wtmpfix	This command corrects the wtmp connect session records that are affected by a date modification and validates login names written to the login name field in the wtmp file.

The /usr/include/utmp.h header file structure is the record format for the following connect session files:

- /var/adm/wtmp
- /var/adm/utmp
- /var/adm/acct/nite/wtmp.*mmdd*
- /var/adm/acct/nite/ctmp

The /usr/include/utmp.h header file structure includes nine fields. Table 13–6 shows the utmp ASCII conversion format for the field number, member name in the header file structure, its description and, if necessary, character length.

Table 13–6: The utmp ASCII Conversion Structure Members

Field	Member	Description
1	ut_user	The user login name, which must have exactly sizeof(ut_user) characters.
2	ut_id	The inittab ID, which must have exactly sizeof(ut_id) characters.
3	ut_line	A memory location, where information used to describe the type of record (for example, the device name) is stored. It must have exactly sizeof(ut_line) characters.
4	ut_pid	The process identification number.

Table 13–6: The utmp ASCII Conversion Structure Members (cont.)

Field	Member	Description
5	ut_type	The type of entry, which can specify several symbolic constant values. The symbolic constants are defined in the /usr/include/utmp.h header file.
6	ut_exit.e_termination	The process termination status.
7	ut_exit.e_exit	The process exit status.
8	ut_time	The starting time (in seconds).
9	ut_host	The host name, which must have exactly sizeof(ut_host) characters.

13.4.1 The wtmpfix Command

The /usr/sbin/acct/wtmpfix command corrects date and time stamp inconsistencies in files with the utmp.h header file structure and displays the records. The runacct script invokes the wtmpfix command.

Each time a date is entered in the /var/adm/wtmp file (for example, at system startup or by using the date command), a pair of date-change records is also written to the wtmp file. The first date-change record is the old date, which is specified in the ut_line and ut_type fields. The second date-change record is the new date, which is also specified in the ut_line and ut_type fields. The wtmpfix command uses these records to synchronize all date and time stamps in the /var/adm/wtmp file, and then the date-change record pair is removed. The date-change records never appear in an output file.

The wtmpfix command also checks the validity of the user name field (the ut_user field) to ensure that the name consists only of alphanumeric characters, a dollar sign ($), or spaces. If an invalid name is detected, the wtmpfix command changes the login name to INVALID and displays a diagnostic message.

The wtmpfix command has the following syntax:

/usr/sbin/acct/wtmpfix [*filename*]...

The *filename* variable specifies the name of the input file. The default input file is the /var/adm/wtmp binary file.

13.4.2 The fwtmp Command

The fwtmp command allows you to correct wtmp files. The command converts binary records from files with the utmp.h header file structure to

formatted ASCII records. You can edit the ASCII version of a wtmp file to repair bad records or for general file maintenance. Table 13–6 shows the ASCII structure you should use.

During system operation, date changes and reboots occur, and the records are written to the /var/adm/wtmp file. The wtmpfix command adjusts the time stamps in the /var/adm/wtmp file; however, some corrections can evade detection by the wtmpfix command and cause the acctcon command to fail. In this case, you can correct the /var/adm/wtmp file by using the fwtmp command.

The fwtmp command has the following syntax:

/usr/sbin/acct/fwtmp [-ic]

The fwtmp file uses standard input, or you can direct a file to the command.

If no options are specified with the fwtmp command, binary records are converted to ASCII records. Refer to the fwtmp(8) reference page for information on command options.

If you want to enter /usr/include/utmp.h header file records manually, you must enter data in each of the nine fields in the order used by the utmp ASCII structure members, as shown in Table 13–6. All record-field entries that you enter from the keyboard must be separated by a space. Also, you must specify all the string fields by using blank characters, if necessary, up to the maximum string size. All decimal values must be specified with the required number of decimal places, using preceding 0s (zeros) to indicate the empty digit positions.

The following example converts the /var/adm/wtmp binary file records to ASCII records:

```
# /usr/sbin/acct/fwtmp  <  /var/adm/wtmp
           system boot  0 20000 0000 652547412 Jan 5 11:10:12 1994
           system boot  0 10062 0123 652547412 Jan 5 11:10:12 1994
bcheck  bl              6 80000 0000 652547413 Jan 5 11:10:13 1994
cat     cr             16 80000 0000 652547414 Jan 5 11:10:14 1994
rc      rc             17 80000 0000 652547485 Jan 5 11:11:25 1994
hoffman co console    147 70000 0001 652547495 Jan 5 11:11:35 1994
hoffman p4 pty/ttyp4 2156 80000 0002 652650095 Jan 6 15:41:35 1994
LOGIN   p4 pty/ttyp4 2140 60000 0000 652649075 Jan 6 15:24:35 1994
LOGIN   p4 pty/ttyp4 2140 80000 0000 652649086 Jan 6 15:24:46 1994
```

To correct a /var/adm/wtmp file:

1. Change your working directory to /var/adm/acct/nite.
2. Use the fwtmp command to create an ASCII version of the wtmp file.

   ```
   # fwtmp < wtmp.0617 > wtmp_temp
   ```

3. Edit the temporary file and remove the corrupted records.

4. Use the fwtmp command to re-create the wtmp file.

```
# fwtmp –ic < wtmp_temp > wtmp.0617
```

13.4.3 The acctwtmp Command

The acctwtmp command allows you to write a reason string and the current time and date to a utmp.h structured file, usually the /var/adm/wtmp file. The runacct, startup, and shutacct shell scripts invoke the acctwtmp command to record when the runacct script is invoked and when system accounting is turned on and off.

The acctwtmp command has the following syntax:

/usr/sbin/acct/acctwtmp *reason*

The *reason* variable must have a maximum of sizeof(ut_line) characters and be enclosed in quotation marks (" ").

13.4.4 The ac Command

The ac command displays connect session records from files with the utmp file structure shown in Table 13–6. You can use the command to perform system diagnostics and determine user charges. The ac command displays the total connect time for all users or the total connect time for the specified users. The connect time is given in hours rounded to the nearest hundredth. To automatically generate total user connect session files, you can include the ac command in the /usr/spool/cron/crontab/adm file or modify the runacct shell script and include the ac command. Refer to Section 13.2.4 for information on setting up automatic accounting.

The ac command has the following syntax:

/usr/sbin/ac [–d] [–p] [–w *filename*] [username...]

Refer to the ac(8) reference page for information on command options.

The default behavior displays the sum of the system connect time for all users. For example:

```
# /usr/sbin/ac
 "total 48804.26"
```

The following command displays the total connect time according to user name:

```
# /usr/sbin/ac  –p
buckler     61.44
fujimori   530.94
```

```
newsnug    122.38
dara         0.10
root       185.98
buchman    339.33
russell     53.96
hoff       200.43
hermi      157.81
total     1968.02
```

The total connect time for all users listed is shown in the last line.

13.4.5 The acctcon1 Command

The `acctcon1` command converts binary session records from a file with the `utmp.h` header file structure to ASCII format. A single record is produced for each connect session. The `runacct` shell script uses the `acctcon1` command to create the `lineuse` and `reboots` files, which are included in the `/var/adm/acct/sum/rprt`*mmdd* daily report.

The `acctcon1` command has the following syntax:

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

You must direct a file as input to the command. Refer to the `acctcon1`(8) reference page for information on command options.

The following command line provides an example of a `/var/adm/acct/nite/lineuse` file. It writes records to the specified file in ASCII line-usage format, which helps you to track line usage and to identify bad lines; and it includes the reference designation of the ports that the user logged in to and the date and time stamp of the currently active connect session.

```
# acctcon1 –l line_file < /var/adm/wtmp | more line_file
TOTAL DURATION IS 57 MINUTES
LINE         MINUTES      PERCENT    # SESS   # ON   # OFF
pty/ttyp4         37           64         3      3       7
console           26           45         2      2       4
pty/ttyp5          7           11         1      1       3
pty/ttyp6          0            0         0      0       2
TOTALS            69            -         6      6      16
```

In the previous example, the ASCII line-usage format specifies the following:

- Total number of minutes that the system was in multiuser state
- The line name
- The number of session minutes used during the accounting period
- The ratio of minutes in use to the total duration

- The number of times the port was accessed (fourth and fifth columns)
- The number of logouts and any other interrupts on the line

You can compare the last column to the fourth column to determine if a line is bad.

The following example produces a sample `/var/adm/acct/reboots` file. It writes records to a file in ASCII overall-record format, which specifies a starting time, an ending time, the number of restarts, and the number of date changes.

```
# acctcon1 –o overall_file < /var/adm/wtmp | more overall_file
from Thu Jan  13 17:20:12 1994 EDT
to   Fri Jan  14 09:56:42 1994 EDT
2   date changes
2   acctg off
0   run-level S
2   system boot
2   acctg on
1   acctcon1
```

The overall-record format includes the `from` and `to` fields, which specify the time that the last accounting report was generated and the time of the current report. These fields are followed by a list of records from the `/var/adm/wtmp` file.

13.4.6 The acctcon2 Command

The `runacct` shell script invokes the `acctcon2` command to convert the `/var/adm/acct/nite/ctmp` connect session file, which is produced by the `acctcon1` command, from ASCII format into binary format.

13.4.7 The prctmp Shell Script

The `prctmp` shell script writes column headings on a connect session database file that has the `utmp.h` header file structure, such as the `/var/adm/acct/nite/ctmp` file, which is created by the `acctcon1` command. The `prctmp` shell script has the following syntax:

/usr/sbin/acct/prctmp [*filename*]

Refer to the `prctmp`(8) reference page for more information.

13.4.8 The lastlogin Shell Script

The `lastlogin` shell script writes the last date that a user logged in to the system to the `/var/adm/acct/sum/loginlog` file. The script invokes the

printpw command to access the login names and user identification numbers in the /etc/passwd file.

The runacct shell script invokes the lastlogin shell script during its CMS state. You can invoke the lastlogin shell script manually to update the /var/adm/acct/sum/loginlog file, which is included in the /var/adm/acct/sum/rprt*mmdd* daily report.

The lastlogin shell script has the following syntax:

/usr/sbin/acct/lastlogin

13.4.9 The last Command

The last command displays, in reverse chronological order, all login records in the /var/adm/wtmp file. For each login session, the following information is provided:

- Time that the session began
- Duration of the session
- tty terminal on which the session took place

The following information is included when applicable:

- Terminations when rebooting
- Continuing sessions

The last command has the following syntax:

/usr/bin/last [-#] [*username*...] [*tty*...]

By default, all records are displayed. You can specify a user name and a terminal for which you want to display records.

The following example displays information only about the three previous root logins:

```
# last -3 root
root    ttyp1     shout     Fri Jan 21 10:56   still logged in
root    ttyp1     raven     Fri Jan 21 08:59 - 09:00  (00:00)
root    ttyp0     raven     Thu Jan 20 15:29 - 15:54  (00:24)
```

13.5 Process Accounting

Process accounting occurs when a command, shell script, or program is executed in the system. When a process exits, the kernel writes the process accounting record to the /var/adm/pacct database file. Process accounting records enable you to monitor program execution statistics. You

can use the ps command to get information about running processes. The accton command creates the /var/adm/pacct file and turns on process accounting.

The /var/adm/pacct file will grow in size. The ckpacct command checks the size of the /var/adm/pacct file and creates a /var/adm/pacct*n* file if the pacct file is larger than a specified size.

The /var/adm/pacct database file includes the following process information:

- Process type (for example, child process)
- Exit status indicating how the process terminated
- User identification number
- Group identification number
- Terminal from which the process originated
- Start, user, system, and CPU time
- Amount of memory used
- Number of I/O characters transferred
- Number of 1024-byte blocks read or written
- Name of the command used to start the process

The record format for the process accounting files is tacct format and is established by the acct header file structure. The acct header file structure is defined in the /usr/include/sys/acct.h header file and includes up to 18 columns of accounting information. The tacct structure members are defined in the private tacct.h header file.

Table 13–7 specifies the column number, heading, and description for files with the tacct format.

Table 13–7: The tacct File Format

Column	Heading	Description
1	UID	Specifies the user identification number, which is obtained from the /etc/passwd file.
2	LOGNAME	Specifies the user login name, which is obtained from the /etc/passwd file.
3	PRI_CPU	Specifies the prime time CPU run time, which is the total time (in seconds) that prime time CPU run time was charged to the user.

Table 13–7: The tacct File Format (cont.)

Column	Heading	Description
4	NPRI_CPU	Specifies the nonprime time CPU run time, which is the total time (in seconds) that nonprime time CPU run time was charged to the user.
5	PRI_MEM	Specifies the prime time memory kcore minutes, which is the total CPU time (in minutes) multiplied by the mean size of the memory used.
6	NPRI_MEM	Specifies the nonprime time memory kcore minutes, which is the total CPU time (in minutes) multiplied by the mean size of the memory used.
7	PRI_RD/WR	Specifies the total number of characters transferred during prime time operation.
8	NPRI_RD/WR	Specifies the total number of characters transferred during nonprime time operation.
9	PRI_BLKIO	Specifies the total number of I/O blocks transferred during prime time read and write operations. The number of bytes in an I/O block depends on how it was implemented.
10	NPRI_BLKIO	Specifies the total number of I/O blocks transferred during nonprime time read and write operations. The number of bytes in an I/O block depends on how it was implemented.
11	PRI_CONNECT	Specifies the total number of prime time seconds that a connection existed.
12	NPRI_CONNECT	Specifies the total number of nonprime time seconds that a connection existed.
13	DSK_BLOCKS	Specifies the total number of disk blocks used.
14	PRINT	Specifies the total number of pages queued to any printer in the system.
15	FEES	Specifies the number of units charged. This value is specified with the `/usr/sbin/acct/chargefee` shell script.
16	PROCESSES	Specifies the total number of processes spawned by the user during the accounting period.

Table 13–7: The tacct File Format (cont.)

Column	Heading	Description
17	SESS	Specifies the total number of times the user logged in during the accounting period.
18	DSAMPS	Specifies the total number of times that the disk accounting command was used to get the total number of disk blocks specified in the DSK_BLOCKS column. You can divide the value in the DSK_BLOCKS column by the value in the DSAMPS column to obtain the average number of disk blocks used during the accounting period.

Process accounting shell scripts and commands allow you to combine information about commands and the resources used to process the commands. The following sections describe the process accounting shell scripts and commands.

13.5.1 The accton Command

The `accton` command enables and disables process accounting. The `accton` command has the following syntax:

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

If you do not specify the *filename* variable, process accounting is disabled. If you specify the *filename* variable, process accounting is turned on and the kernel writes process accounting records to the specified file. Usually, this file is the `/var/adm/pacct` file; however, you can specify a different process accounting database file. The file must exist in the `/var/adm` directory, be owned by user `adm`, and be a member of the `adm` login group.

Note

The `runacct` and `turnacct` shell scripts use the `/var/adm/pacct` process accounting database file. If you specify a process accounting database file other than the `/var/adm/pacct` file, the `runacct` and `turnacct` shell scripts will be affected.

13.5.2 The turnacct Shell Script

The `turnacct` shell script controls the process accounting functions and creates process accounting files. You must be superuser to use the shell script. The `turnacct` script has the following syntax:

turnacct [on|off|switch]

The `turnacct on` shell script turns on process accounting by invoking the `accton` shell script with the `/var/adm/pacct` file argument.

The `turnacct off` shell script turns off process accounting by invoking the `accton` command without an argument to disable process accounting.

The `turnacct switch` shell script moves the contents of the `/var/adm/pacct` file to the `/var/adm/pacct`*n* file and then creates a new `/var/adm/pacct` file.

13.5.3 The ckpacct Shell Script

The `/var/adm/pacct` file can grow in size. If the `/var/adm/pacct` file is larger than a specified limit and if enough disk space is available, the `ckpacct` script invokes the `turnacct switch` shell script to move the contents of the `/var/adm/pacct` file to the `/var/adm/pacct`*n* file and create a new `/var/adm/pacct` file.

You can set up your `cron` daemon to invoke the `ckpacct` script periodically. Refer to Section 13.2.4 for more information.

The `ckpacct` shell script has the following syntax:

ckpacct [*blocksize*]

The *blocksize* variable specifies the size limit (in disk blocks) for the `/var/adm/pacct` file. The default size is 500 disk blocks.

If you invoke the `ckpacct` shell script, the script checks the number of disk blocks that are available in the `/var/adm` directory. If the number of available blocks is less than the size limit, process accounting is disabled by invoking the `turnacct off` shell script. A diagnostic message is displayed and mailed to the address that is specified with the MAILCOM environment variable. Use the `putenv` function to set the MAILCOM environment variable to the following command:

```
mail root adm
```

The following diagnostic message shows that there are 224 disk blocks remaining in the `/var/adm` directory:

```
ckpacct: /var/adm too low on space  (224 blocks)
  "turning acctg off"
```

The `ckpacct` shell script continues to display diagnostic messages until adequate space exists in the `/var/adm` directory.

13.5.4 The acctcom Command

The acctcom command displays summaries of process accounting records. Command options allow you to specify the type and format of the output. You do not have to be superuser to use the acctcom command.

The acctcom command displays information only about processes that have terminated; use the ps command to display information about active processes. The acctcom command has the following syntax:

/usr/bin/acctcom [*option*...] [*filename*...]

If you do not specify the *filename* variable, the command uses the /var/adm/pacct file to obtain the process accounting records. You can use the *filename* variable to specify a different process accounting file that has the acct.h header file structure. If you specify more than one *filename* variable, the acctcom command reads the files in chronological order.

If you do not specify any command options, the default output includes the following information in a column heading format:

- Time and date that accounting was enabled
- Command name
- User name
- tty name
- Process start time
- Process end time
- Real seconds
- CPU seconds
- Mean memory size (in kilobytes)

Refer to the acctcom(8) reference page for information on the command options.

The following is an example of the default process accounting summary output:

```
# /usr/bin/acctcom  /var/adm/pacct1
ACCOUNTING RECORDS FROM: Mon Jan 17 02:00:00 1994
COMMAND                  START    END        REAL   CPU    MEAN
NAME     USER   TTYNAME  TIME     TIME      (SECS) (SECS) SIZE(K)
#sa      root   ttyp1    11:59:00 11:59:00   0.77   0.01   0.00
ls       root   ttyp1    11:59:04 11:59:04   0.11   0.01   0.00
uugetty  root   ?        11:58:39 11:59:48  69.53   0.01   0.00
#ls      root   ttyp1    11:59:55 11:59:55   0.30   0.01   0.00
uugetty  root   ?        11:59:49 12:00:58  69.48   0.01   0.00
```

```
cp        adm     ?         12:05:01 12:05:01   0.33   0.01   0.00
chmod     adm     ?         12:05:01 12:05:01   0.27   0.01   0.00
#df       adm     ?         12:05:02 12:05:02   0.38   0.01   0.00
awk       adm     ?         12:05:02 12:05:02   0.58   0.01   0.00
sed       adm     ?         12:05:02 12:05:02   0.56   0.01   0.00
```

13.5.5 The sa Command

The sa command summarizes process accounting information. This command helps you to manage the large volume of accounting information. The files produced by the sa command include all the available process accounting information. The sa command has the following syntax:

/usr/sbin/sa [*options...*] [*filename*]

The *filename* variable specifies a process accounting file with the acct.h header file structure. If the *filename* variable is not specified, the /var/adm/pacct file is used.

If you invoke the sa command with no options, the default output consists of six unheaded columns. Certain command options allow you to expand the six columns to include more information. You can specify options to change the format and to output additional information that includes an identifying suffix. Refer to the sa(8) reference page for information on the command options.

The following example shows the default format of the output of the sa command:

```
# /usr/sbin/sa
798      277.24re       0.08cpu    3248790avio      0k
  7       33.42re       0.08cpu     103424avio      0k    csh
 14        0.08re       0.00cpu     127703avio      0k    mv
 40        0.34re       0.00cpu     159968avio      0k    cp
  2        0.01re       0.00cpu     132448avio      0k    acctwtmp
 34        0.13re       0.00cpu     133517avio      0k    chmod
 23        0.10re       0.00cpu     139136avio      0k    chgrp
 25        0.11re       0.00cpu     144768avio      0k    chown
 36        0.15re       0.00cpu     133945avio      0k    dspmsg
 32        0.18re       0.00cpu     134206avio      0k    cat
[1]          [2]          [3]          [4]          [5]    [6]
```

[1] Shows information about the number of command executions. An additional column is added to show the command percentage if you specify the –c option.

[2] Shows information about the amount of real time used. An additional column is added to show the real-time percentage if you specify the –c option.

[3] Shows information about CPU time used. Depending on the options specified, the column can show the total system and user CPU time, the user CPU time, the system CPU time, or the ratio of user CPU

time to system CPU time. An additional column is added to show the real-time percentage if you specify the –c option. Also, an additional column is added to show the ratio of real time to total user and system CPU time if you specify the –t option.

4 Shows information about disk I/O operations, either the average number of I/O operations or the total number of I/O operations.

5 Shows information about kiloblocks (number of blocks multiplied by 1024) used or the memory time integral.

6 Shows the command name.

The following example adds three columns to the default format to display the following percentages:

```
# /usr/sbin/sa -c
645 100.00% 324.10re 100.00% 0.02cpu 100.00% 6171050avio 0k
  2   0.31%  25.70re   7.93% 0.02cpu 100.00%  107392avio 0k csh
  6   0.93%   0.04re   0.01% 0.00cpu   0.00%  132928avio 0k mv
 38   5.89%   0.33re   0.10% 0.00cpu   0.00%  163357avio 0k cp
  2   0.31%   0.01re   0.00% 0.00cpu   0.00%  132992avio 0k cat
 26   4.03%   0.11re   0.03% 0.00cpu   0.00%  136832avio 0k chmod
 24   3.72%   0.10re   0.03% 0.00cpu   0.00%  139824avio 0k chgrp
      1           2             3
```

The additional columns show the following information:

1 Indicates the number of times each command was executed with respect to the total number of times all commands were executed.

2 Indicates the amount of real time needed to execute the command the number of times specified in column one with respect to the total real time required to execute all the commands.

3 Indicates the amount of CPU time needed to execute the command the number of times specified in column 1 with respect to the total CPU time required to execute all commands.

13.5.6 The acctcms Command

The `acctcms` command produces ASCII and binary total command summary files from process accounting records. You specify process accounting files that have the `/usr/include/sys/acct.h` header file structure, such as the `/var/adm/pacct` file. The `acctcms` command sorts the records and combines the statistics for each command used during the accounting period into a single record. The records allow you to identify the commands used most and the commands that use the most system time.

The `runacct` shell script invokes the `acctcms` command during its CMS state. You can also invoke this command manually to create a command summary report.

The acctcms command has the following syntax:

/usr/sbin/acct/acctcms [-acjnopst] *filename...*

If you invoke the acctcms command with no options, the command sorts the output in descending order according to total kcore minutes, which is the number of kilobytes of memory used by the process multiplied by the buffer time used. Binary output is the default. Use the following calculation to obtain the kcore minutes:

```
kcoremin=[(CPU time in seconds)*(mean memory size in kbyte)]/60
```

Refer to the acctcms(8) reference page for information on the command options.

Note

If you use the acctcms command to produce a total summary file in ASCII format, each command record will consist of more than 80 characters, and the entire width of 8.5 x 11-inch paper could be used if the 10-character per inch constant-width font is specified. If part of a record exceeds the column width, it is moved to the next line.

The following example produces ASCII output that includes the statistics for commands that were invoked only once in a row specifying ***other in the COMMAND NAME column:

```
# acctcms -a -j /var/adm/pacct1
                        TOTAL COMMAND SUMMARY
COMMAND NUMBER TOTAL  TOTAL  TOTAL   MEAN  MEAN   HOG    CHARS  BLOCKS
NAME    CMDS KCOREMIN CPUMIN REALMIN SIZEK CPUMIN FACTOR TRNSFD READ

TOTALS  9377  0.00    0.36  26632.67 0.00  0.00   0.00 17768213 100529

chmod     34  0.00    0.00       .15 0.00  0.00   0.07  5785856    64
ln         4  0.00    0.00      0.01 0.00  0.00   0.78   422016    16
xterm      9  0.00    0.03    537.41 0.00  0.00   0.00 22948288   536
getcons    8  0.00    0.00      0.14 0.00  0.00   0.07 26636992   102
cfe2.20    4  0.00    0.00      0.09 0.00  0.00   0.12   182464   155
dump      22  0.00    0.00     14.91 0.00  0.00   0.00 69402112   128
whoami     4  0.00    0.00      0.03 0.00  0.00   0.36  7405952    27
restore   40  0.00    0.00     49.16 0.00  0.00   0.00 34247488  1316
***other  25  0.00    0.00   3546.88 0.00  0.00   0.00 35904984   737
hostname   2  0.00    0.00      0.01 0.00  0.01   0.94   223104    14
```

The hog factor is the total CPU time divided by the total real time.

13.5.7 The acctprc1 Command

The acctprc1 command reads process accounting records from files with the /usr/include/sys/acct.h header file structure, adds the login

names that correspond to the user identification numbers, and displays the records in ASCII format. Login session records are sorted according to user identification number and login name.

If your system has users with the same user identification number, you should use a process accounting file in the `/var/adm/acct/nite` directory instead of the `/var/adm/pacct` file.

The `runacct` shell script invokes the `acctprc1` command during its PROCESS state. You can also invoke the command manually. The `acctprc1` command has the following syntax:

/usr/sbin/acct/acctprc1 [*filename*]

The *filename* variable specifies a file that contains a list of login sessions in a format defined by the `/usr/include/utmp.h` header file structure. If the *filename* variable is not specified, login names are obtained from the `/etc/passwd` file.

The command output specifies information in a format with seven unheaded columns that specify the following:

- User identification number
- Login name
- Number of CPU seconds the process used during prime time
- Number of CPU seconds the process used during nonprime time
- Total number of characters transferred
- Total number of blocks read from and written to
- Mean memory size (in kilobytes)

The following is an example of the `acctprc1` command and its output:

```
# /usr/sbin/acct/acctprc1  < /usr/adm/pacct
   0   root      0     1   17228   172    6
   4   adm       0     6   46782    46   16
   0   root      0    22  123941   132   28
9261   hoffmann  6     0   17223    22   20
   9   lp        2     0   20345    27   11
9261   hoffmann  0   554   16554    20  234
```

13.5.8 The acctprc2 Command

The `acctprc2` command reads records produced by the `acctprc1` command, summarizes them according to user identification number and login name, and then uses the `tacct` file format to display the sorted summaries as total accounting binary records. You can merge the binary

file produced by the acctprc2 command with other total accounting files by using the acctmerg command to produce a daily summary accounting record file.

The runacct shell script invokes the acctprc2 command during its PROCESS state. You can also invoke the command manually.

13.5.9 The lastcomm Command

The lastcomm command displays command execution information from the /var/adm/pacct file in reverse chronological order.

The following information is displayed for each process:

- Command name
- Either the S flag, which specifies that the command was invoked by the superuser; or the F flag, which specifies that the command ran after a fork but was not followed by an exec system call
- Name of the user who issued the command
- Terminal from which the command was started
- Number of seconds of CPU time used
- Time the process started

The lastcomm command has the following syntax:

/usr/bin/lastcomm [*command*] [*username*] [*tty*]

The following example displays information about the sed commands executed by root:

```
# lastcomm sed root
sed      S  root     ttyp0      0.01 secs Fri Jan 21 11:34
sed      S  root     ttyp0      0.01 secs Fri Jan 21 11:34
```

13.6 Disk Usage Accounting

Disk usage accounting is performed by the dodisk shell script. The dodisk shell script uses either the diskusg or the acctdusg command to write information to the intermediate ASCII file /var/adm/dtmp. The shell script then uses the intermediate file as input to the acctdisk command to create a binary total accounting database file, /var/adm/acct/nite/dacct. The dodisk script performs disk accounting on all or selected file systems specified in the /etc/fstab file system database file.

You can combine the total accounting information in the /var/adm/acct/nite/dacct file with other accounting information to create complete accounting reports. For example:

```
# /usr/sbin/acct/dodisk
# /usr/sbin/acct/prtacct /var/adm/acct/nite/dacct
```

13.6.1 The dodisk Shell Script

Use the dodisk shell script to obtain disk usage accounting. You can set up your cron daemon to run the dodisk script automatically, or you can invoke the command manually. The dodisk shell script has the following syntax:

/usr/sbin/acct/dodisk [–o] [*filesystem*...]

/usr/sbin/acct/dodisk [*device special file*...]

Using the –o option, you can specify the file system variable to perform disk usage accounting on the mount point of a UFS file system or an AdvFS fileset. If the –o option is not specified, the variable must be the raw or character device special file. For example:

```
# /usr/sbin/acct/dodisk /dev/rrz3c
```

If you do not specify any arguments, disk accounting is performed on the UFS device special files decribed in the /etc/fstab database file. Refer to the fstab(4) reference page for more information.

Note

If you have a swap space specified in the /etc/fstab file, the dodisk shell script will not execute correctly. In this case, you can edit the dodisk shell script to use only specific file systems or you can invoke the dodisk shell script and specify the file systems for which you want accounting.

If you specify the –o option, the dodisk shell script uses the acctdusg command instead of the diskusg command to perform a more thorough but slower version of disk accounting. If you specify the –o option and a *filesystem* variable, specify the mount point instead of the device special file name.

13.6.2 The diskusg Command

The diskusg command displays disk accounting records. The diskusg command obtains user login names and identification numbers from the /etc/passwd file. The diskusg command has the following syntax:

/usr/sbin/acct/diskusg [–options] [*filesystems*]

Refer to the diskusg(8) reference page for information on the command options.

The diskusg command produces ASCII output, which is directed to the /var/adm/dtmp file. This file is used as input to the acctdisk command, which converts the ASCII records to binary total accounting records in the /var/adm/acct/nite/dacct file. You can merge these records with other accounting records to create a daily total accounting report.

Each output record produced by the diskusg command contains the user identification number, login name, and the total number of disk blocks allocated to the user. Because the diskusg command checks user inode records, all disk space is accounted for, including empty directories.

The following is an example of the diskusg command:

```
# /usr/sbin/acct/diskusg  /dev/rrz3c
   0   root        63652
   1   daemon         84
   2   bin         71144
   4   adm           976
   5   uucp         3324
 322   homer           2
 521   whistler        2
 943   cellini       363
1016   pollock        92
1098   hopper        317
```

You must specify the raw device special file for filesystem (for example, /dev/rrz3c). A file system must exist on the target device.

13.6.3 The acctdusg Command

The acctdusg command performs more thorough disk accounting than the diskusg command. If dodisk is invoked with the –o option, the acctdusg command is used to create the /var/adm/dtmp file.

The acctdusg command has the following syntax:

acctdusg [–u *filename*] [–p *filename*]

Refer to the acctdusg(8) reference page for information on the command options.

You must direct a binary disk usage file, usually /var/adm/dtmp, to the command. If the dodisk shell script invokes the command, the acctdusg command uses the file systems specified with the dodisk script as input.

The input to the acctdusg command is usually a list of files piped from a find / –print command. The command compares the file pathnames to the users' login directories ($HOME). If a file pathname is the same as a user's login directory, that user is charged for the file. Therefore, the directory in which the file is located is the determining factor in charging users for disk space. You can use the –u option to display the number of disk blocks used by files in directories other than the login directories.

For each file, the acctdusg command calculates the computed value, which is the number of disk blocks (including hidden or indirect blocks) that are allocated to the file divided by the number of hard links. If two or more users have links to the same file, the acctdusg command charges each user an equal percentage of the file's total disk space.

The acctdusg command output displays the user identification number, the user name, and the sum of the computed values of all the files owned by the user in three columns and adds leading 0s (zeros) to the user identification number. The acctdusg command does not display the disk-block count for empty directories.

13.6.4 The acctdisk Command

The acctdisk command creates a binary total accounting file. If it is invoked from the dodisk script, the acctdisk command reads the /var/adm/dtmp file that is produced by either the diskusg or acctdusg command. It then writes converted binary records to a temporary file, which is then moved to the /var/adm/acct/nite/dacct file.

The disk usage accounting records produced by the acctdisk command are usually merged with other accounting records to produce a total accounting report.

13.7 System Administration Service Accounting

You can charge users for system administration services. For example, you could charge for the following services:

- Backing up files to disk
- Recovering files from disk
- Backing up files to tape
- Recovering files from tape
- Providing software technical assistance by phone
- Providing software technical assistance in person

The chargefee shell script allows you to charge users according to the work performed. You should determine how much you want to charge for each service. Services can have different charge rates according to the time it takes to perform the task.

Charge units are collected in the /var/adm/fee file. You can use the number of units charged to a user name to determine the fees for the system administration tasks. The chargefee shell script creates the /var/adm/fee file, if necessary, and adds a record that includes the user identification number, user name, and charge units.

The chargefee shell script has the following syntax:

/usr/sbin/acct/chargefee *user_name units*

You can subtract units by specifying a dash (–) with the *units* variable.

The following example charges 7 units to user josh:

```
# chargefee josh 7
```

If the previous command is issued, the following record is written to the /var/adm/fee file:

```
1114 josh 0 0 0 0 0 0 0 0 0 0 0 0 7 0 0 0
```

13.8 Printer Accounting

When you use a printer that has accounting enabled, a record is written to the printer accounting file. Printer accounting records have a specific syntax and provide the following information:

- Name of the host and user that issued the print request
- Number of pages or feet of medium printed
- Number of times the printer was used
- Price per unit of printed output

The printer accounting records enable you to charge users for the system printing resources and to track printer usage.

The two printer accounting files are located in either the /var/adm or the /var/adm/printer directory. The *printer.*acct printer user file lists the amount and cost of print media used, according to machine and user name. The *printer.*acct_sum printer summary file lists a summary of media produced according to machine and user name. The *printer* variable specifies the printer name. Refer to Chapter 11 for information on creating the printer accounting files.

Use the pac command to create a report of your printer activity. The pac command can obtain information only for printers that have accounting enabled. The pac command has the following syntax:

pac [-cmrs] [-p *price*] [-P *printer*] [*user*...]

Refer to the pac(8) reference page for information on the command options.

13.9 Creating Daily, Summary, and Monthly Report Files

There are four shell scripts and one command that you can use to create daily, summary, and monthly report files in the /var/adm/acct/nite, /var/adm/acct/sum, and /var/adm/acct/fiscal directories, as shown in the following table:

Command	Description
runacct	This shell script creates the daily and summary files in the /var/adm/acct/nite and /var/adm/acct/sum directories.
acctmerg	This command merges total accounting record files and allows you to combine process connect time, fee, disk usage, and print queue accounting records into files whose format you specify. The output can be in either the default binary format or ASCII format and can include up to 18 columns of accounting information.
prtacct	This shell script formats and displays accounting files that have the /usr/include/sys/acct.h header file structure. Each record includes information about the user identification number, connect time, process time, disk usage, and printer usage.
prdaily	This shell script creates an ASCII file that contains the accounting data from the previous day. When this script is invoked from the runacct script, it creates the /var/adm/acct/sum/rprt*mmdd* file.
monacct	This shell script creates cumulative process and total accounting files in the /var/adm/acct/fiscal directory.

The following sections describe the shell scripts and the command in detail.

13.9.1 The runacct Shell Script

The runacct shell script uses accounting shell scripts and commands to process the connect time, fee, disk usage, queue, and process accounting

database files to create the daily and summary files in the /var/adm/acct/nite and /var/adm/acct/sum directories.

The /var/adm/acct/nite directory contains files that are reused daily by the runacct script. Some of these files have binary counterparts in the /var/adm/acct/sum directory, which contains the cumulative summary files that are updated by the runacct shell script and used by the monacct shell script to produce monthly reports.

You can set up the cron daemon to invoke the runacct shell script each day, or you can invoke the runacct shell script manually. You may have to invoke the command manually if the runacct shell script does not complete or if a file created by the script becomes corrupted or lost.

Note

When you invoke the runacct shell script it creates the /var/adm/acct/nite/lock temporary file. If the /var/adm/acct/nite/lock file exists, the runacct shell script will not run.

The runacct shell script executes in the following 13 states, in the order listed, and can be restarted at any of the 13 states:

State	Description
SETUP	Sets up some of the accounting files.
WTMPFIX	Fixes corrupted date and time stamp entries that can cause commands such as the acctcon1 command to fail.
CONNECT1	Writes connect session records.
CONNECT2	Uses the connect session records to create a binary total accounting record that will be merged with other records to create a daily report.
PROCESS	Produces process accounting report files.
MERGE	Uses the acctmerg command to create the binary total accounting file.
FEES	Uses the acctmerg command to merge records from the /var/adm/fee file into the binary total accounting file.
DISK	Uses the acctmerg command to merge disk-usage records into the binary total accounting file.

State	Description
QUEUEACCT	Uses the `acctmerg` command to merge print queue accounting records into the binary total accounting file.
MERGEACCT	Copies the binary total accounting file to the daily total accounting file, which is used as input to the `acctmerg` command to create the cumulative total daily accounting file.
CMS	Produces command usage summaries.
USEREXIT	Invokes any site-specific shell scripts.
CLEANUP	Removes the temporary files.

13.9.1.1 Correcting runacct Shell Script Errors

If a `runacct` shell script error occurs, a message is written to the console device, the lock file is removed, the diagnostic files and error messages are saved, and processing is halted. Use the following information to determine if a `runacct` shell script error has occurred:

- The `/var/adm/acct/nite/active` file is created if the script has successfully completed. The `runacct` shell script logs messages to this file. You can use this file to determine which tasks have been successfully completed. The following is an example of an `active` file:

```
Fri Feb 4 11:02:56 EST 1994
-rw-r--r-- 1 adm  adm      0 Jan 31 03:00 /var/adm/acct/nite/dacct
-rw-rw-r-- 1 root system 924 Jan 05 10:45 /var/adm/wtmp
-rw-rw-r-- 1 adm  adm      0 Jan 08 13:46 fee
-rw-rw-r-- 1 adm  adm      0 Jan 07 02:00 pacct
-rw-rw-r-- 1 adm  adm   8904 Jan 02 11:02 pacct1
files setups complete
wtmp processing complete
connect acctg complete
process acctg complete for /var/adm/Spacct1.1101
process acctg complete for /var/adm/Spacct2.1101
all process acctg complete for 1101
tacct merge to create daytacct complete
no fees
no disk records
no queueing system records
updated sum/tacct
command summaries complete
system accounting completed at Fri
```

- The `/var/adm/acct/nite/active`*`mmdd`* file is created if the script has not successfully completed. This file contains information about the script execution; you can use it to determine where the script failed.
- The `/var/adm/acct/nite/statefile` file contains the name of the last state that the `runacct` shell script executed. Note that the `runacct` shell script may not have successfully completed this state.

- The /var/adm/acct/nite/lastdate file contains the date of the last runacct shell script execution. If the date specified in the file is the current date, the shell script will not run.

If the runacct shell script fails or terminates before it is completed, you must restart the script from its last successfully completed state. The /var/adm/acct/nite/statefile file contains the name of the state that was last executed.

The runacct shell script has the following syntax:

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

The *mmdd* variable specifies the date for which you want to run the runacct shell script. Use the *state* variable to specify the state from which you want the runacct script to start processing.

If the runacct shell script fails on more than one successive day, invoke the SETUP state commands manually.

Note

Before you restart the runacct shell script, you should remove the /var/adm/acct/nite/lock file and the /var/adm/acct/nite/lastdate file.

In the following example, the runacct shell script is invoked at its MERGE state and uses the accounting database files from January 26:

```
# runacct 0126 MERGE > /var/adm/nite/fd2log&
```

The following example invokes the runacct shell script, which uses the accounting database files from January 26 and specifies the nohup command so that signals, hangups, logouts, and quits are disregarded; any error messages generated during its execution are written to the fd2log file:

```
# nohup runacct 0126 > /var/adm/acct/nite/fd2log&
```

13.9.1.2 Examples of Errors and Corrective Actions

The following list provides examples of errors and the actions you can take to correct problems:

ERROR: locks found. run aborted
: A /var/adm/acct/nite/lock file exists. Remove the file and restart the runacct shell script from its last completed state.

`ERROR: acctg already run for Fri : check Jan`
The current date is the same as the date specified in the `/var/adm/acct/nite/lastdate` file. Remove the file and restart the `runacct` shell script from its last completed state.

`ERROR: runacct called with invalid arguments`
You have specified invalid arguments with the `runacct` shell script.

`ERROR: turnacct switch returned rc=?`
The `accton` command failed when it was invoked by the `turnacct switch` shell script. Check the `accton` command protections and ensure that user `adm` can invoke the command.

`ERROR: Spacct?.mmdd already exists run setup manually`
You must invoke the `runacct` shell script manually from the MERGE state.

`ERROR: wtmpfix errors see nite/wtmperror`
An unrepairable `wtmp` file was found during the WTMPFIX state. Use the `fwtmp` command to correct the file.

`ERROR: invalid state, check /usr/var/adm/nite/active`
During processing, the `runacct` shell script may have detected a corrupted `active` file. Check the `/var/adm/acct/nite/active*` and `statefile` files.

13.9.2 The acctmerg Command

The `acctmerg` command combines process, connect time, fee, disk-usage, and queue total accounting record files with the `tacct` file format. For example, you can merge the total accounting records for a particular login name and user identification number to provide a single group of records for that login name and user identification number. File records are usually merged according to the user identification number or the user login name.

The default command output is in binary format, but you can also produce ASCII output. The default `acctmerg` command output has the `/usr/include/sys/acct.h` header file structure and includes up to 18 columns of accounting information. Records with the `/usr/include/sys/acct.h` header file structure that include data types specified as an array of two double elements can have both prime time and nonprime time values.

The runacct shell script invokes the acctmerg command. You can also invoke the command manually to produce reports. The acctmerg command has the following syntax:

/usr/sbin/acct/acctmerg [-ahiptuv] [*#*] [*file*...]

You can specify up to nine total accounting record files. If you do not specify a file, records are read from standard input.

Refer to the acctmerg(8) reference page for information on command options.

The following example reads the UID, LOGNAME, DSK_BLOCKS, and DSAMPS column entries from the /var/adm/acct/nite/dacct ASCII disk accounting file. It then merges them into binary records in the /var/adm/acct/sum/tacct total accounting file.

```
# acctmerg  -i1-2, 13, 18 < nite/dacct | sum/tacct
```

You can use the acctmerg command to correct errors in the /var/adm/sum/tacct file. Errors that can occur in the file include negative numbers and duplicate user identification numbers.

To correct errors in the current /var/adm/sum/tacct file:

1. Change your directory to /var/adm/sum.
2. Enter the prtacct command to display the /var/adm/sum/tacctprev file. If the file is correct, then the problem probably is located in the /var/adm/sum/tacct*mmdd* file. This example assumes that the /var/adm/sum/tacct*mmdd* file needs to be fixed.
3. To obtain an ASCII version of the /var/adm/sum/tacct*mmdd* file, enter:

   ```
   # acctmerg –v < tacct.0617 > tacct_temp
   ```
4. Edit the temporary file and correct the records as necessary.
5. To re-create the /var/adm/sum/tacct*mmdd* file, enter:

   ```
   # acctmerg –i < tacct_temp > tacct.0617
   ```
6. To re-create the /var/adm/sum/tacct file, enter:

   ```
   # acctmerg tacctprev < tacct.0617 > tacct
   ```

13.9.3 The prtacct Shell Script

The prtacct shell script displays a binary total accounting file with the tacct file format in ASCII format. The script allows you to produce a connect time, process time, disk usage, or printer usage report file.

The monacct and prdaily shell scripts invoke the prtacct shell script. The runacct shell script invokes the prdaily shell script during its CLEANUP state. The prtacct shell script has the following syntax:

/usr/sbin/acct/prtacct [–f *column*] [–v] *file*

Refer to the prtacct(8) reference page for information on the command options.

13.9.4 The prdaily Shell Script

The prdaily shell script creates an ASCII report of the accounting data from the previous day. The runacct shell script invokes the prdaily shell script during its CLEANUP state to create the /var/adm/acct/sum/rprt*mmdd* file. You can invoke the command manually to produce a report.

The prdaily script combines information from the following six accounting files:

- /var/adm/acct/nite/reboots
- /var/adm/acct/nite/lineuse
- /var/adm/acct/sum/tacct*mmdd*
- /var/adm/acct/nite/daycms
- /var/adm/acct/nite/cms
- /var/adm/acct/sum/loginlog

The prdaily shell script has the following syntax:

prdaily [-l[*mmdd*]]|[-c]

Refer to prdaily(8) for more information on command options.

13.9.5 The monacct Shell Script

The monacct shell script uses the binary accounting files to create cumulative summary files in the /var/adm/acct/fiscal directory. After the summary files are produced, the command removes the old accounting files from the /var/adm/acct/sum directory and creates new files.

Usually, you run the monacct script once each month to produce monthly report files. You can set up your cron daemon to run the shell script automatically. Refer to Section 13.2.4 for more information. The monacct shell script has the following syntax:

/usr/sbin/acct/monacct [*number*]

The *number* variable specifies an integer that is within the range 1 to 12 and that specifies the month for which you want to create the summary report. The default is the current month.

The `monacct` shell script creates the following files in the `/var/adm/acct/fiscal` directory:

`tacct`*mm*	Specifies the binary total accounting file for the month preceding the month specified by the *mm* variable.
`cms`*mm*	Specifies the binary cumulative command summary file for the month preceding the month specified by the *mm* variable.
`fiscrpt`*mm*	Specifies the ASCII total monthly accounting report file. This file has a format that is similar to the `/var/adm/acct/sum/rprt`*mmdd* report file and is created from the following files: • `/var/adm/acct/fiscal/tacct`*mm* • `/var/adm/acct/fiscal/cms`*mm* • `/var/adm/acct/sum/loginlog`

14

Administering Events and Errors

This chapter first describes how to use the system exercizers to discover potential system problems. Then, the chapter describes how the Digital UNIX operating system records information about system events and explains the basic administrative tasks that you use to set up and maintain the event-logging system.

14.1 Using the System Exercisers

The Digital UNIX system provides a set of exercisers that you can use to troubleshoot your system. The exercisers test specific areas of your system, such as file systems or system memory. This chapter explains how to use the exercisers by addressing the following topics:

- Running the system exercisers (Section 14.1.1)
- Using exerciser diagnostics (Section 14.1.2)
- Exercising file systems by using the `fsx` command (Section 14.1.3)
- Exercising system memory by using the `memx` command (Section 14.1.4)
- Exercising shared memory by using the `shmx` command (Section 14.1.5)
- Exercising disk drives by using the `diskx` command (Section 14.1.6)
- Exercising tape drives by using the `tapex` command (Section 14.1.7)
- Exercising communications systems by using the `cmx` command (Section 14.1.8)

In addition to the exercisers documented in this chapter, your system might also support the DEC Verifier and Exerciser Tool (VET), which provides a similar set of exercisers. With the release of Digital UNIX Version 4.0, VET is no longer present on the installation kit as an optional subset. Instead, VET software is on the Digital UNIX Firmware CD-ROM.

14.1.1 Running System Exercisers

To run a system exerciser, you must be logged in as superuser and `/usr/field` must be your current directory.

The commands that invoke the system exercisers provide a flag for specifying a file where diagnostic output is saved when the exerciser completes its task.

Most of the exerciser commands have an online help flag that displays a description of how to use that exerciser. To access online help, use the –h flag with a command. For example, to access help for the diskx exerciser, use the following command:

```
# diskx –h
```

The exercisers can be run in the foreground or the background and can be canceled at any time by pressing Ctrl/C in the foreground. You can run more than one exerciser at the same time; keep in mind, however, that the more processes you have running, the slower the system performs. Thus, before exercising the system extensively, make sure that no other users are on the system.

There are some restrictions when you run a system exerciser over an NFS link or on a diskless system. For exercisers such as fsx that need to write to a file system, the target file system must be writable by root. Also, the directory from which an exerciser is executed must be writable by root because temporary files are written to the directory.

These restrictions can be difficult to adhere to because NFS file systems are often mounted in a way that prevents root from writing to them. Some of the restrictions may be adhered to by copying the exerciser into another directory and then executing it.

14.1.2 Using Exerciser Diagnostics

When an exerciser is halted (either by pressing Ctrl/C or by timing out), diagnostics are displayed and are stored in the exerciser's most recent log file. The diagnostics inform you of the test results.

Each time an exerciser is invoked, a new log file is created in the /usr/field directory. For example, when you execute the fsx command for the first time, a log file named #LOG_FSX_01 is created. The log files contain records of each exerciser's results and consist of the starting and stopping times, and error and statistical information. The starting and stopping times are also logged into the default system error log file, /var/adm/binary.errlog. This file also contains information on errors reported by the device drivers or by the system.

The log files provide a record of the diagnostics. However, after reading a log file, you should delete it because an exerciser can have only nine log files. If you attempt to run an exerciser that has accumulated nine log files, the exerciser tells you to remove some of the old log files so that it can create a new one.

If an exerciser finds errors, you can determine which device or area of the system has the difficulty by looking at the /var/adm/binary.errlog file,

using either the dia command (preferred) or the uerf command. For information on the error logger, see the Section 14.2. For the meanings of the error numbers and signal numbers, see the intro(2) and sigvec(2) reference pages.

14.1.3 Exercising a File System

Use the fsx command to exercise the local file systems. The fsx command exercises the specified local file system by initiating multiple processes, each of which creates, writes, closes, opens, reads, validates, and unlinks a test file of random data. For more information, see the fsx(8) reference page.

Note

Do not test NFS file systems with the fsx command.

The fsx command has the following syntax:

fsx [-f*path*] [–h] [–o*file*] [–p*num*] [–t*min*]

You can specify one or more of the following flags:

–f*path*	Specifies the pathname of the file system directory you want to test. For example, –f/usr or –f/mnt. The default is /usr/field.
–h	Displays the help message for the fsx command.
–o*file*	Saves the output diagnostics in *file.*
–p*num*	Specifies the number of fsxr processes you want fsx to initiate. The maximum number of processes is 250. The default is 20.
–t*min*	Specifies how many minutes you want the fsx command to exercise the file system. If you do not specify the –t flag, the fsx command runs until you terminate it by pressing Ctrl/C in the foreground.

The following example of the fsx command tests the /usr file system with five fsxr processes running for 60 minutes in the background:

```
# fsx -p5 -f/usr -t60 &
```

14.1.4 Exercising System Memory

Use the memx command to exercise the system memory. The memx command exercises the system memory by initiating multiple processes. By default, the size of each process is defined as the total system memory in bytes divided by 20. The minimum allowable number of bytes per process is 4095. The memx command runs 1s and 0s, 0s and 1s, and random data patterns in the allocated memory being tested.

The files that you need to run the memx exerciser include the following:

- memx
- memxr

For more information, see the memx(8) reference page

The memx command is restricted by the amount of available swap space. The size of the swap space and the available internal memory determine how many processes can run simultaneously on your system. For example, if there are 16 MB of swap space and 16 MB of memory, all of the swap space will be used if all 20 initiated processes (the default) run simultaneously. This would prevent execution of other process. Therefore, on systems with large amounts of memory and small amounts of swap space, you must use the -p or -m flag, or both, to restrict the number of memx processes or to restrict the size of the memory being tested.

The memx command has the following syntax:

memx -s [-h] [-m*size*] [-o*file*] [-p*num*] [-t*min*]

You can specify one or more of the following flags:

-s	Disables the automatic invocation of the shared memory exerciser, shmx.
-h	Displays the help message for the memx command.
-m*size*	Specifies the amount of memory in bytes for each process you want to test. The default is the total amount of memory divided by 20, with a minimum size of 4095 bytes.
-o*file*	Saves the output diagnostics in *file.*
-p*num*	Specifies the number of memxr processes to initiate. The maximum number is 20, which is also the default.

-t*min* Specifies how many minutes you want the memx command to exercise the memory. If you do not specify the -t flag, the memx command runs until you terminate it by pressing Ctrl/C in the foreground.

The following example of the memx command initiates five memxr processes that test 4095 bytes of memory and runs in the background for 60 minutes:

```
# memx –m4095 –p5 –t60 &
```

14.1.5 Exercising Shared Memory

Use the shmx command to exercise the shared memory segments. The shmx command spawns a background process called shmxb. The shmx command writes and reads the shmxb data in the segments, and the shmxb process writes and reads the shmx data in the segments.

Using shmx, you can test the number and the size of memory segments and shmxb processes. The shmx exerciser runs until the process is killed or until the time specified by the –t flag is exhausted.

You automatically invoke the shmx exerciser when you start the memx exerciser, unless you specify the memx command with the –s flag. You can also invoke the shmx exerciser manually. The shmx command has the following syntax:

/usr/field/shmx [-h] [-o*file*] [-v] [-t*time*] [-m*size*] [-s*n*]

The shmx command flags are as follows:

–h Prints the help message for the shmx command.

–o*file* Saves diagnostic output in *file*.

–v Uses the fork system call instead of the vfork system call to spawn the shmxb process.

-t*time* Specifies *time* as the run time in minutes. The default is to run until the process is killed.

-m*size* Specifies *size* as the memory segment size, in bytes, to be tested by the processes. The *size* value must be greater than zero. The default is the value of the SHMMAX and SHMSEG system parameters,

which are set in the /sys/include/sys/param.h file.

`-s`*n* — Specifies *n* as the number of memory segments. The default (and maximum) number of segments is 3.

The following example tests the default number of memory segments, each with a default segment size:

```
# shmx &
```

The following example runs three memory segments of 100,000 bytes for 180 minutes:

```
# shmx -t180 -m100000 -s3 &
```

14.1.6 Exercising a Disk Drive

Use the diskx command to exercise the disk drives. The main areas that are tested include the following:

- Reads, writes, and seeks
- Performance
- Disktab entry verification

Caution

Some of the tests involve writing to the disk; for this reason, use the exerciser cautiously on disks that contain useful data that the exerciser could overwrite. Tests that write to the disk first check for the existence of file systems on the test partitions and partitions that overlap the test partitions. If a file system is found on these partitions, you are prompted to determine if testing should continue.

You can use the diskx command flags to specify the tests that you want performed and to specify the parameters for the tests. For more information, see the diskx(8) reference page.

The diskx command has the following syntax:

diskx [*flags*] [*parameters*] -f *devname*

The -f *devname* flag specifies the device special file on which to perform testing. The *devname* variable specifies the name of the block or character

special file that represents the disk to be tested. The file name must begin with an r (for example, rz1). The last character of the file name can specify the disk partition to test.

If a partition is not specified, all partitions are tested. For example, if the *devname* variable is /dev/rra0, all partitions are tested. If the *devname* variable is /dev/rra0a, the a partition is tested. This parameter must be specified and can be used with all test flags.

The following flags specify the tests to be run on disk:

–d
Tests the disk's disktab file entry. The disktab entry is obtained by using the getdiskbyname library routine. This test only works if the specified disk is a character special file. See the disktab(4) reference page for more information.

–h
Displays a help message describing test flags and parameters.

–p
Specifies a performance test. Read and write transfers are timed to measure device throughput. Data validation is not performed as part of this test. Testing uses a range of transfer sizes if the –F flag is not specified.

The range of transfer sizes is divided by the number specified with the perf_splits parameter to obtain a transfer size increment. For example, if the perf_splits parameter is set to 10, tests are run starting with the minimum transfer size and increasing the transfer size by 1/10th of the range of values for each test repetition. The last transfer size is set to the specified maximum transfer size.

If you do not specify a number of transfers, the transfer count is set to allow the entire partition to be read or written. In this case, the transfer count varies, depending on the transfer size and the partition size.

The performance test runs until completed or until interrupted; the time is not limited by the –minutes parameter. This test can take a long time to complete, depending on the test parameters.

To achieve maximum throughput, specify the –S flag to cause sequential transfers. If the –S flag is not specified, transfers are done to random locations. This may slow down the observed throughput because of associated head seeks on the device.

–r

Specifies a read-only test. This test reads from the specified partitions. Specify the –n flag to run this test on the block special file.

This test is useful for generating system I/O activity. Because it is a read-only test, you can run more than one instance of the exerciser on the same disk.

–w

Specifies a write test. This test verifies that data can be written to the disk and can be read back to verify the data. Seeks are also done as part of this test. This test provides the most comprehensive coverage of disk transfer functions because it uses reads, writes, and seeks. This test also combines sequential and random access patterns.

This test performs the following operations using a range of transfer sizes; a single transfer size is used if the –F attribute is specified:

- Sequentially writes the entire test partition, unless the number of transfers has been specified using the –num_xfer parameter
- Sequentially reads the test partition

The data read from the disk is examined to verify it. Then, if random transfer testing has not been disabled (using the –S attribute), writes are issued to random locations on the partition. After the random writes are completed, reads are issued to random locations on the partition. The data read from random locations is examined to verify it.

The following flags modify the behavior of the test:

–F

Performs fixed size transfers. If this flag is not specified, transfers are done using random sizes. This flag can be used with the –p, –r, and –w test flags.

–i

Specifies interactive mode. In this mode, you are prompted for various test parameters. Typical parameters include the transfer size and the number of transfers. The following scaling factors are allowed:

- k or K (for kilobyte (1024 * n))
- b or B (block (512 * n))
- m or M (megabyte (1024 * 1024 * n))

For example 10K would specify 10,240 bytes.

–Q
: Suppresses performance analysis of read transfers. This flag only performs write performance testing. To perform only read testing and to skip the write performance tests, specify the –R flag. The –Q flag can be used with the –p test flag.

–R
: Opens the disk in read-only mode. This flag can be used with all test flags.

–S
: Performs transfers to sequential disk locations. If this flag is not specified, transfers are done to random disk locations. This flag can be used with the –p, –r, and –w test flags.

–T
: Directs output to the terminal. This flag is useful if output is directed to a log file by using the –o flag. If you specify the –T flag after the –o flag, output is directed to both the terminal and the log file. The –T flag can be used with all test flags.

–Y
: Does not prompt you to confirm that you want to continue the test if file systems are found when the disk is examined; testing proceeds.

In addition to the flags, you can also specify test parameters. You can specify test parameters on the `diskx` command line or interactively with the –i flag. If you do not specify test parameters, default values are used.

To use a parameter, specify the parameter name, a space, and the numeric value. For example, you could specify the following parameter:

```
–perf_min 512
```

You can use the following scaling factors:

- k or K (for kilobyte (1024 * n))
- b or B (for block (512 * n))
- m or M (for megabyte (1024 * 1024 * n))

For example, 10K would specify 10,240 bytes, and `–perf_min 10K` causes transfers to be done in sizes of 10,240 bytes.

You can specify one or more of the following parameters:

–debug
: Specifies the level of diagnostic output to be produced. The greater the number specified, the more output is produced describing the exerciser

operations. This parameter can be used with all test flags.

–err_lines	Specifies the maximum number of error messages that are produced as a result of an individual test. A limit on error output prevents a large number of diagnostic messages if persistent errors occur. This parameter can be used with all test flags.
–minutes	Specifies the number of minutes to test. This parameter can be used with the –r and –w test flags.
–max_xfer	Specifies the maximum transfer size to be performed. If transfers are done using random sizes, the sizes are within the range specified by the –max_xfer and –min_xfer parameters. If fixed size transfers are specified (see the –F flag), transfers are done in a size specified by the –min_xfer parameter. Specify transfer sizes to the character special file in multiples of 512 bytes. If the specified transfer size is not an even multiple, the value is rounded down to the nearest 512 bytes. This parameter can be used with the –r and –w test flags.
–min_xfer	Specifies the minimum transfer size to be performed. This parameter can be used with the –r and –w test flags.
–num_xfer	Specifies the number of transfers to perform before changing the partition that is currently being tested. This parameter is only useful if more than one partition is being tested. If this parameter is not specified, the number of transfers is set to a number that completely covers a partition. This parameter can be used with the –r and –w test flags.
–o*filename*	Sends output to the specified file name. The default is to display output on the terminal screen. This parameter can be used with all test flags.

–perf_max	Specifies the maximum transfer size to be performed. If transfers are done using random sizes, the sizes are within the range specified by the –perf_min and –perf_max parameters. If fixed size transfers are specified (see the –F flag), transfers are done in a size specified by the –perf_min parameter. This parameter can be used with the –p test flag.
–perf_min	Specifies the minimum transfer size to be performed. This parameter can be used with the –p test flag.
–perf_splits	Specifies how the transfer size will change if you test a range of transfer sizes. The range of transfer sizes is divided by the number specified with the –perf_splits parameter to obtain a transfer size increment. For example, if the –perf_splits parameter is set to 10, tests are run starting with the minimum transfer size and increasing the transfer size by 1/10th of the range of values for each test repetition. The last transfer size is set to the specified maximum transfer size. This parameter can be used with the –p test flag.
–perf_xfers	Specifies the number of transfers to be performed in performance analysis. If this value is not specified, the number of transfers is set equal to the number that is required to read the entire partition. This parameter can be used with the –p test flag.

The following example performs read-only testing on the character device special file that /dev/rrz0 represents. Because a partition is not specified, the test reads from all partitions. The default range of transfer sizes is used. Output from the exerciser program is displayed on the terminal screen:

```
# diskx -f /dev/rrz0 -r
```

The following example runs on the a partition of /dev/rz0, and program output is logged to the diskx.out file. The program output level is set to 10 and causes additional output to be generated:

```
# diskx -f /dev/rz0a -o diskx.out -d -debug 10
```

The following example shows that performance tests are run on the a partition of /dev/rz0, and program output is logged to the diskx.out file. The –S flag causes sequential transfers for the best test results. Testing is done over the default range of transfer sizes:

```
# diskx -f /dev/rz0a -o diskx.out -p -S
```

The following command runs the read test on all partitions of the specified disks. The disk exerciser is invoked as three separate processes, which generate extensive system I/O activity. The command shown in this example can be used to test system stress:

```
# diskx -f /dev/rrz0 -r &; diskx -f /dev/rrz1 -r &; diskx -
f /dev/rrz2 -r &
```

14.1.7 Exercising a Tape Drive

Use the tapex command to exercise a tape drive. The tapex command writes, reads, and validates random data on a tape device from the beginning-of-tape (BOT) to the end-of-tape (EOT). The tapex command also performs positioning tests for records and files, and tape transportability tests. For more information, refer to the tapex(8) reference page.

Some tapex flags perform specific tests (for example, an end-of-media (EOM) test). Other flags modify the tests, for example, by enabling caching.

The tapex command has the following syntax:

tapex [*flags*] [*parameters*]

You can specify one or more of the flags described in Table 14–1. In addition to flags, you can also specify test parameters. You specify parameters on the tapex command line or interactively with the –i flag. If you do not specify test parameters, default values are used.

To use a test parameter, specify the parameter name, a space, and the number value. For example, you could specify the following parameter:

```
–min_rs 512
```

Note that you can use the following scaling factors:

- k or K (for kilobyte (1024 * n))
- b or B (for block (512 * n))
- m or M (for megabyte (1024 * 1024 * n))

For example, 10K would specify 10,240 bytes.

The following parameters can be used with all tests:

–err_lines Specifies the error printout limit.

–fixed *bs* Specifies a fixed block device. Record sizes for most devices default to multiples of the blocking factor of the fixed block device as specified by the *bs* argument.

The following parameters can be used with the –a flag, which measures performance:

–perf_num Specifies the number of records to write and read.

–perf_rs Specifies the size of records.

Other parameters are restricted for use with specific tapex flags. Option-specific parameters are documented in Table 14–1.

Table 14–1: The tapex Options and Option Parameters

tapex Flag	Flag and Parameter Descriptions
–a	Specifies the performance measurement test, which calculates the tape transfer bandwidth for writes and reads to the tape by timing data transfers. The following parameters can be used with the –a flag: –perf_num — Specifies the number of records to write and read. –perf_rs — Specifies the size of records.
–b	Causes the write/read tests to run continuously until the process is killed. This flag can be used with the –r and –g flags.
–c	Enables caching on the device, if supported. This flag does not specifically test caching; it enables the use of caching on a tape device while other tests are running.
–C	Disables caching on TMSCP tape devices. If the tape device is a TMSCP unit, then caching is the default mode of test operation. This flag causes the tests to run in noncaching mode.

Table 14–1: The tapex Options and Option Parameters (cont.)

tapex Flag	Flag and Parameter Descriptions	
–d	Tests the ability to append records to the media. First, the test writes records to the tape. Then, it repositions itself back one record and appends additional records. Finally, the test does a read verification. This test simulates the behavior of the `tar –r` command. The following parameters can be used with the –d flag:	
	–no_overwrite	Prevents the append-to-media test from being performed on tape devices that do not support this test. Usually, you use this parameter with the –E flag.
	–tar_num	Specifies the number of additional and appended records.
	–tar_size	Specifies the record size for all records written in this test.
–e	Specifies EOM test. First, this test writes data to fill a tape; this action can take some time for long tapes. It then performs reads and writes past the EOM; these actions should fail. Finally, it enables writing past the EOM, writes to the tape, and reads back the records for validation purposes.	
	The following parameters can be used with the –e flag:	
	–end_num	Specifies the number or records to be written past EOM. (Note that specifying too much data to be written past EOM can cause a reel-to-reel tape to go off line.)
	–end_rs	Specifies the record size.
–E	Runs an extensive series of tests in sequential order. Depending on tape type and CPU type, this series of tests can take up to 10 hours to complete.	
–f /dev/rmtl*#?*	Specifies the name of the device special file that corresponds to the tape unit being tested. The number sign variable (*#*) specifies the unit number. The question mark variable (*?*) specifies the letter `h` for the high density device or `l` for the low density device. The default tape device is `/dev/rmt0h`.	

Table 14–1: The tapex Options and Option Parameters (cont.)

tapex Flag	Flag and Parameter Descriptions
–F	Specifies the file-positioning tests. First, files are written to the tape and verified. Next, every other file on the tape is read. Then, the previously unread files are read by traversing the tape backwards. Finally, random numbers are generated, the tape is positioned to those locations, and the data is verified. Each file uses a different record size. The following parameters can be used with the –F flag: –num_fi — Specifies the number of files. –pos_ra — Specifies the number of random repositions. –pos_rs — Specifies the record size. –rec_fi — Specifies the number of records per file.
–G	Specifies the file-positioning tests on a tape containing data. This flag can be used with the –F flag to run the file position tests on a tape that has been written to by a previous invocation of the –F test. To perform this test, you must use the same test parameters (for example, record size and number of files) that you used when you invoked the –F test to write to the tape. No other data should have been written to the tape since the previous –F test.
–g	Specifies random record size tests. This test writes records of random sizes. It reads in the tape, specifying a large read size; however, only the amount of data in the randomly sized record should be returned. This test only checks return values; it does not validate record contents. The following parameter is used with the –g flag: –rand_num — Specifies the number of records to write and read.
–h	Displays a help message describing the tape exerciser.
–i	Specifies interactive mode. In this mode, you are prompted for various test parameters. Typical parameters include the record size and the number of records to write. The following scaling factors are allowed: • k or K (for kilobyte (1024 * n)) • b or B (for block (512 * n)) • m or M (for megabyte (1024 * 1024 * n)) For example, 10K would specify 10,240 bytes.

Table 14–1: The tapex Options and Option Parameters (cont.)

tapex Flag	Flag and Parameter Descriptions
–j	Specifies the write phase of the tape-transportability tests. This test writes a number of files to the tape and then verifies the tape. After the tape has been successfully verified, it is brought off line, moved to another tape unit, and read in with the –k flag. This test proves that you can write to a tape on one drive and read from a tape on another drive. The –j flag is used with the –k flag. Note the –j flag and the –k flag must use the same parameters. The following parameters can be used with the –j and –k flags: –tran_file — Specifies the number of files to write or read. –tran_rec — Specifies the number of records contained in each file. –tran_rs — Specifies the size of each record.
–k	Specifies the read phase of the tape-transportability tests. This test reads a tape that was written by the –j test and verifies that the expected data is read from the tape. This test proves that you can write to a tape on one drive and read from a tape on another drive. As stated in the description of the –j flag, any parameters specified with the –j flag must be specified with the –k flag. (See the description of the –j flag for information on the parameters that apply to the –j and –k flags.)
–L	Specifies the media loader test. For sequential stack loaders, the media is loaded, written to, and verified. Then, the media is unloaded, and the test is run on the next piece of media. This verifies that all of the media in the input deck can be written to. To run this test in read-only mode, also specify the –w flag.
–l	Specifies the EOF test. This test verifies that a zero byte count is returned when a tape mark is read and that an additional read fetches the first record of the next tape file.
–m	Displays tape contents. This is not a test. This flag reads the tape sequentially and prints out the number of files on the tape, the number of records in each file, and the size of the records within the file. The contents of the tape records are not examined.
–o filename	Sends output to the specified file name. The default sends output to the terminal screen.
–p	Runs both the record-positioning and file-positioning tests. For more information, refer to descriptions of the –R and –F flags.

Table 14–1: The tapex Options and Option Parameters (cont.)

tapex Flag	Flag and Parameter Descriptions	
–q	Specifies the command timeout test. This test verifies that the driver allows enough time for completion of long operations. This test writes files to fill the tape. It then performs a rewind, followed by a forward skip to the last file. This test is successful if the forward skip operation is completed without error.	
–r	Specifies the record size test. A number of records are written to the tape and then verified. This process is repeated over a range of record sizes. The following parameters can be used with the –r flag:	
	–inc	Specifies the record increment factor.
	–max_rs	Specifies the maximum record size.
	–min_rs	Specifies the minimum record size.
	–num_rec	Specifies the number of records.
	–t	Specifies a time limit (in minutes). The default is to run the test until it is complete.
–R	Specifies the record-positioning test. First, records are written to the tape and verified. Next, every other record on the tape is read. Then, the other records are read by traversing the tape backwards. Finally, random numbers are generated; the tape is positioned to those locations, and the data is verified. The following parameters can be used with the -R flag:	
	–pos_num	Specifies the number of records.
	–pos_ra	Specifies the number of random repositions.
	–pos_rs	Specifies the record size.
–s	Specifies the record size behavior test. Verifies that a record that is read returns one record (at most) or the read size, whichever is less. The following parameters can be used with the –s flag:	
	–num_rec	Specifies the number of records.
	–size_rec	Specifies the record size.

Table 14–1: The tapex Options and Option Parameters (cont.)

tapex Flag	Flag and Parameter Descriptions
–S	Specifies single record size test. This test modifies the record size test (the –r flag) to use a single record size. The following parameters can be used with the -S flag:
	–inc — Specifies the record increment factor.
	–max_rs — Specifies the maximum record size.
	–min_rs — Specifies the minimum record size.
	–num_rec — Specifies the number of records.
–T	Displays output to the terminal screen. This flag is useful if you want to log output to a file with the –o flag and also have the output displayed on your terminal screen. This flag must be specified after the –o flag in the command line.
–v	Specifies verbose mode. This flag causes detailed information to be output. For example, it lists the operations the exerciser is performing (such as record counts), and detailed error information. Information provided by this flag can be useful for debugging purposes.
–V	Specifies enhanced verbose mode. This flag causes output of more detailed information than the –v flag. The additional output consists of status information on exerciser operations. Information provided by this flag can be useful for debugging purposes.
–w	Opens the tape as read-only. This mode is useful only for tests that do not write to the media. For example, it allows the –m test to be run on a write-protected media.
–Z	Initializes the read buffer to the nonzero value 0130. This can be useful for debugging purposes. If the –Z flag is not specified, all elements of the read buffer are initialized to zero. Many of the tests first initialize their read buffer and then perform the read operation. After reading a record from the tape, some tests validate that the unused portions of the read buffer remain at the value to which they were initialized. For debugging purposes, you can set this initialized value to a number other than zero. In this case, you can use the arbitrary value 0130.

The following example runs an extensive series of tests on tape device `rmt1h` and sends all output to the `tapex.out` file:

```
# tapex –f /dev/rmt1h -E -o tapex.out
```

The following example performs random record size tests and outputs information in verbose mode. This test runs on the default tape device /dev/rmt0h, and the output is sent to the terminal screen.

```
# tapex -g -v
```

The following example performs read and write record testing using record sizes in the range 10K to 20K. This test runs on the default tape device /dev/rmt0h, and the output is sent to the terminal screen.

```
# tapex -r -min_rs 10k -max_rs 20k
```

The following example performs a series of tests on tape device /dev/rmt0h, which is treated as fixed block device in which record sizes for tests are multiples of the blocking factor 512 KB. The append-to-media test is not performed.

```
# tapex -f /dev/rmt0h -fixed 512 -no_overwrite
```

14.1.8 Exercising the Terminal Communication System

Use the cmx command to exercise the terminal communications system. The cmx command writes, reads, and validates random data and packet lengths on the specified communications lines.

The lines you exercise must have a loopback connector attached to the distribution panel or the cable. Also, the line must be disabled in the /etc/inittab file and in a nonmodem line; that is, the CLOCAL flag must be set to on. Otherwise, the cmx command repeatedly displays error messages on the terminal screen until its time expires or until you press Ctrl/C. For more information, refer to the cmx(8) reference page.

You cannot test pseudodevice lines or lta device lines. Pseudodevices have p, q, r, s, t, u, v, w, x, y, or z as the first character after tty, for example, ttyp3.

The cmx command has the following syntax:

/usr/field/cmx [-h] [-o *file*] [-t *min*] [-l *line*]

The cmx command flags are as follows:

–h	Prints a help message for the cmx command.
–o *file*	Saves output diagnostics in *file*.
–t *min*	Specifies how many minutes you want the cmx command to exercise the communications system. If

you do not specify the –t flag, the cmx command runs until you terminate it by pressing Ctrl/C in the foreground.

–l *line* — Specifies the line or lines you want to test. The possible values for *line* are found in the /dev directory and are the last two characters of the tty device name. For example, if you want to test the communications system for devices named tty02, tty03, and tty14, specify 02, 03, and 14, separated by spaces, for the *line* variable. In addition, the line variable can specify a range of lines to test. For example, 00-08.

The following example exercises communication lines tty22 and tty34 for 45 minutes in the background:

```
# cmx -l 22 34 -t45 &
```

The following example exercises lines tty00 through tty07 until you press Ctrl/C:

```
# cmx -l 00-07
```

14.2 Understanding the Event-Logging Facilities

The Digital UNIX operating system uses two mechanisms to log system events:

- The system event-logging facility
- The binary event-logging facility

The log files that the system and binary event-logging facilities create have the default protection of 640, are owned by root, and belong to the system group. You must have the proper authority to examine the files.

The following sections describe the event-logging facilities.

14.2.1 System Event Logging

The primary systemwide event-logging facility uses the syslog function to log events in ASCII format. The syslog function uses the syslogd daemon to collect the messages that are logged by the various kernel, command, utility, and application programs. The syslogd daemon logs the messages to a local file or forwards the messages to a remote system, as specified in the /etc/syslog.conf file.

When you install your Digital UNIX operating system, the `/etc/syslog.conf` file is created and specifies the default event-logging configuration. The `/etc/syslog.conf` file specifies the file names that are the destination for the event messages, which are in ASCII format. Section 14.3.1.1 discusses the `/etc/syslog.conf` file.

14.2.2 Binary Event Logging

The binary event-logging facility detects hardware and software events in the kernel and logs the detailed information in binary format records. Events that are logged by the binary event-logging facility are also logged by the `syslog` function in a less detailed, but still informative, summary message.

The binary event-logging facility uses the `binlogd` daemon to collect various event-log records. The `binlogd` daemon logs these records to a local file or forwards the records to a remote system, as specified in the `/etc/binlog.conf` default configuration file, which is created when you install your Digital UNIX system.

With Digital UNIX Version 4.0, the event management utility of choice is the DECevent component, in place of the `uerf` error logging facility. You can examine the binary event-log files by using the `dia` command (preferred) or by using the `uerf` command. Both commands translate the records from binary format to ASCII format.

Note

The `uerf` facility remains as a component of Digital UNIX, but will be retired in a future release of the operating system. See Appendix D or `uerf`(8) for more information about using `uerf`.

The DECevent utility is an event managment utility that you can use to produce ASCII reports from entries in the system's event log files. The DECevent utility can be used from the command line and it can be run by selecting it from the System Management Utilities menu box.

For information about administering the `DECevent` utility, see the following Digital UNIX documentation:

- *DECevent Translation and Reporting Guide*
- `dia`(8)

Section 14.3.1.2 discusses the `/etc/binlog.conf` file.

14.3 Configuring Event Logging

When you install your system, the default system and binary event-logging configuration is used. You can change the default configuration by modifying the configuration files. You can also modify the binary event-logging configuration, if necessary.

To enable system and binary event-logging, the special files must exist and the event-logging daemons must be running. Refer to Section 14.3.2 and Section 14.3.3 for more information.

14.3.1 Editing the Configuration Files

If you do not want to use the default system or binary event-logging configuration, edit the `/etc/syslog.conf` or `/etc/binlog.conf` configuration file to specify how the system should log events. In the files, you specify the facility, which is the source of a message or the part of the system that generates a message; the priority, which is the message's level of severity; and the destination for messages.

The following sections describe how to edit the configuration files.

14.3.1.1 The syslog.conf File

If you want the `syslogd` daemon to use a configuration file other than the default, you must specify the file name with the `syslogd –f` *`config_file`* command.

The following is an example of the default `/etc/syslog.conf` file:

```
#
# syslogd config file
#
# facilities: kern user mail daemon auth syslog lpr binary
# priorities: emerg alert crit err warning notice info debug
#
# 1    2                              3
kern.debug              /var/adm/syslog.dated/kern.log
user.debug              /var/adm/syslog.dated/user.log
daemon.debug            /var/adm/syslog.dated/daemon.log
auth.crit;syslog.debug  /var/adm/syslog.dated/syslog.log
mail,lpr.debug          /var/adm/syslog.dated/misc.log
msgbuf.err              /var/adm/crash.dated/msgbuf.savecore
kern.debug              /var/adm/messages
kern.debug              /dev/console
*.emerg                 *
```

Each /etc/syslog.conf file entry has the following entry syntax:

1. Specifies the facility, which is the part of the system generating the message.
2. Specifies the severity level. The syslogd daemon logs all messages of the specified severity level plus all messages of greater severity. For example, if you specify level err, all messages of levels err, crit, alert, and emerg or panic are logged.
3. Specifies the destination where the messages are logged.

The syslogd daemon ignores blank lines and lines that begin with a number sign (#). You can specify a number sign (#) as the first character in a line to include comments in the /etc/syslog.conf file or to disable an entry.

The facility and severity level are separated from the destination by one or more tabs.

You can specify more than one facility and its severity level by separating them with semicolons. In the preceding example, messages from the auth facility of crit severity level and higher and messages from the syslog facility of debug severity level and higher are logged to the /var/adm/syslog.dated/syslog.log file.

You can specify more than one facility by separating them with commas. In the preceding example, messages from the mail and lpr facilities of debug severity level and higher are logged to the /var/adm/syslog.dated/misc.log file.

You can specify the following facilities:

Facility	Description
kern	Messages generated by the kernel. These messages cannot be generated by any user process.
user	Messages generated by user processes. This is the default facility.
mail	Messages generated by the mail system.
daemon	Messages generated by the system daemons.
auth	Messages generated by the authorization system (for example: login, su, and getty).
lpr	Messages generated by the line printer spooling system (for example: lpr, lpc, and lpd).
local0	Reserved for local use, along with local1 to local7.
mark	Receives a message of priority info every 20 minutes, unless a different interval is specified with the syslogd –m option.

Facility	Description
msgbuf	Kernel syslog message buffer recovered from a system crash. The savecore command and the syslogd daemon use the msgbuf facility to recover system event messages from a crash.
*	Messages generated by all parts of the system.

You can specify the following severity levels, which are listed in order of highest to lowest severity:

Severity Level	Description
emerg or panic	A panic condition. You can broadcast these messages to all users.
alert	A condition that you should immediately correct, such as a corrupted system database.
crit	A critical condition, such as a hard device error.
err	Error messages.
warning or warn	Warning messages.
notice	Conditions that are not error conditions, but are handled as special cases.
info	Informational messages.
debug	Messages containing information that is used to debug a program.
none	Disables a specific facility's messages.

You can specify the following message destinations:

Destination	Description
Full pathname	Appends messages to the specified file. You should direct each facility's messages to separate files (for example: kern.log, mail.log, or lpr.log).
Host name preceded by an at sign (@)	Forwards messages to the syslogd daemon on the specified host.
List of users separated by commas	Writes messages to the specified users if they are logged in.
*	Writes messages to all the users who are logged in.

You can specify in the /etc/syslog.conf file that the syslogd daemon create daily log files. To create daily log files, use the following syntax to specify the pathname of the message destination:

/var/adm/syslog.dated/ {*file*}

The *file* variable specifies the name of the log file, for example, mail.log or kern.log.

If you specify a /var/adm/syslog.dated/*file* pathname destination, each day the syslogd daemon creates a subdirectory under the /var/adm/syslog.dated directory and a log file in the subdirectory by using the following syntax:

/var/adm/syslog.dated/ *date* / *file*

The *date* variable specifies the day, month, and time that the log file was created.

The *file* variable specifies the name of the log file you previously specified in the /etc/syslog.conf file.

The syslogd daemon automatically creates a new *date* directory every 24 hours and also when you boot the system.

For example, to create a daily log file of all mail messages of level info or higher, edit the /etc/syslog.conf file and specify an entry similar to the following:

```
mail.info  /var/adm/syslog.dated/mail.log
```

If you specify the previous command, the syslogd daemon could create the following daily directory and file:

```
/var/adm/syslog.dated/11-Jan-12:10/mail.log
```

14.3.1.2 The binlog.conf File

If you want the binlogd daemon to use a configuration file other than the default, specify the file name with the binlogd -f *config_file* command.

The following is an example of a /etc/binlog.conf file:

```
#
# binlogd configuration file
#
# format of a line:   event_code.priority         destination
#
# where:
# event_code - see codes in binlog.h and man page, * = all events
```

```
# priority   - severe, high, low, * = all priorities
# destination - local file pathname or remote system hostname
#
#
*.*   /usr/adm/binary.errlog
dumpfile  /usr/adm/crash/binlogdumpfile
102.high  /usr/adm/disk.errlog
1    2                    3
```

Each entry in the `/etc/binlog.conf` file, except the `dumpfile` event class entry, contains three fields:

1. Specifies the event class code that indicates the part of the system generating the event.
2. Specifies the severity level of the event. Do not specify a severity level if you specify `dumpfile` for an event class.
3. Specifies the destination where the binary event records are logged.

The `binlogd` daemon ignores blank lines and lines that begin with a number sign (#). You can specify a number sign (#) as the first character in a line to include comments in the file or to disable an entry.

The event class and severity level are separated from the destination by one or more tabs.

You can specify the following event class codes:

Class Code	General
*	All event classes.
`dumpfile`	Specifies the recovery of the kernel binary event log buffer from a crash dump. A severity level cannot be specified.

Class Code	Hardware-Detected Events
100	CPU machine checks and exceptions
101	Memory
102	Disks
103	Tapes
104	Device controller
105	Adapters
106	Buses
107	Stray interrupts
108	Console events

Class Code	Hardware-Detected Events
109	Stack dumps
199	SCSI CAM events

Class Code	Software-Detected Events
201	CI port-to-port-driver events
202	System communications services events

Class Code	Informational ASCII Messages
250	Generic

Class Code	Operational Events
300	Startup ASCII messages
301	Shutdown ASCII messages
302	Panic messages
310	Time stamp
350	Diagnostic status messages
351	Repair and maintenance messages

You can specify the following severity levels:

Severity Level	Description
*	All severity levels
`severe`	Unrecoverable events that are usually fatal to system operation
`high`	Recoverable events or unrecoverable events that are not fatal to system operation
`low`	Informational events

You can specify the following destinations:

Destination	Description
Full pathname	Specifies the file name to which the binlogd daemon appends the binary event records.
@*hostname*	Specifies the name of the host (preceded by an @) to which the binlogd daemon forwards the binary event records. If you specify dumpfile for an event class, you cannot forward records to a host.

14.3.2 Creating the Special Files

The syslogd daemon cannot log kernel messages unless the /dev/klog character special file exists. If the /dev/klog file does not exist, create it by using the following command syntax:

/dev/MAKEDEV /dev/klog

Also, the binlogd daemon cannot log local system events unless the /dev/kbinlog character special file exists. If the /dev/kbinlog file does not exist, create it by using the following command syntax:

/dev/MAKEDEV /dev/kbinlog

Refer to the MAKEDEV(8) reference page for more information.

14.3.3 Starting and Stopping the Event-Logging Daemons

The syslogd and binlogd daemons are automatically started by the init program during system startup. However, you must ensure that the daemons are started. You can also specify options with the command that starts the daemons. Refer to the init(8) reference page for more information.

14.3.3.1 The syslogd Daemon

You must ensure that the syslogd daemon is started by the init program. If the syslogd daemon is not started or if you want to specify options with the command that starts the syslogd daemon, you must edit the /sbin/init.d/syslog file and either include or modify the syslogd command line. Note that you can also invoke the command manually.

The command that starts the syslogd daemon has the following syntax:

/usr/sbin/syslogd [-d] [-f *config_file*] [-m *mark_interval*]

Refer to the syslogd(8) reference page for information about command options.

Note

You must ensure that the /var/adm directory is mounted, or the syslogd daemon will not work correctly.

The syslogd daemon reads messages from the following:

- The Digital UNIX domain socket /dev/log file, which is automatically created by the syslogd daemon
- An Internet domain socket, which is specified in the /etc/services file
- The special file /dev/klog, which logs only kernel messages

Messages from other programs use the openlog, syslog, and closelog calls.

When the syslogd daemon is started, it creates the /var/run/syslog.pid file, where the syslogd daemon stores its process identification number. Use the process identification number to stop the syslogd daemon before you shut down the system.

During normal system operation, the syslogd daemon is called if data is put in the kernel syslog message buffer, located in physical memory. The syslogd daemon reads the /dev/klog file and gets a copy of the kernel syslog message buffer. The syslogd daemon starts at the beginning of the buffer and sequentially processes each message that it finds. Each message is prefixed by facility and priority codes, which are the same as those specified in the /etc/syslog.conf file. The syslogd daemon then sends the messages to the destinations specified in the file.

To stop the syslogd event-logging daemon, use the following command:

```
# kill 'cat /var/run/syslog.pid'
```

You can apply changes that you make to the /etc/syslog.conf configuration file without shutting down the system by using the following command:

```
# kill -HUP 'cat /var/run/syslog.pid'
```

14.3.3.2 The binlogd Daemon

You must ensure that the init program starts the binlogd daemon. If the binlogd daemon does not start, or if you want to specify options with the command that starts the binlogd daemon, you must edit the /sbin/init.d/syslog file and either include or modify the binlogd command line. Note that you can also invoke the command manually.

The command that starts the binlogd daemon has the following syntax:

/usr/sbin/binlogd [-d] [-f *config_file*]

Refer to the binlogd(8) reference page for information on command options.

The binlogd daemon reads binary event records from the following:

- An Internet domain socket (binlogd, 706/udp), which is specified in the /etc/services file
- The /dev/kbinlog special file

When the binlogd daemon starts, it creates the /var/run/binlogd.pid file, where the binlogd daemon stores its process identification number. Use the process identification number to stop or reconfigure the binlogd daemon.

During normal system operation, the binlogd daemon is called if data is put into the kernel's binary event-log buffer or if data is received on the Internet domain socket. The binlogd daemon then reads the data from the /dev/kbinlog special file or from the socket. Each record contains an event class code and a severity level code. The binlogd daemon processes each binary event record and logs it to the destination specified in the /etc/binlog.conf file.

To stop the binlogd daemon, use the following command:

```
# kill 'cat /var/run/binlogd.pid'
```

You can apply changes that you make to the /etc/binlog.conf configuration file without shutting down the system by using the following command:

```
# kill -HUP 'cat /var/run/binlogd.pid'
```

14.3.4 Configuring the Kernel Binary Event Logger

You can configure the kernel binary event logger by modifying the default keywords and rebuilding the kernel. You can scale the size of the kernel binary event-log buffer to meet your systems needs. You can enable and disable the binary event logger and the logging of kernel ASCII messages into the binary event log.

The /sys/data/binlog_data.c file defines the binary event-logger configuration. The default configuration specifies a buffer size of 24K bytes, enables binary event logging, and disables the logging of kernel ASCII messages. You can modify the configuration by changing the values of the binlog_bufsize and binlog_status keywords in the file.

The binlog_bufsize keyword specifies the size of the kernel buffer that the binary event logger uses. The size of the buffer can be between 8 kilobytes (8192 bytes) and 48 kilobytes (49152 bytes). Small system configurations, such as workstations, can use a small buffer. Large server systems that use many disks may need a large buffer.

The binlog_status keyword specifies the behavior of the binary event logger. You can specify the following values for the binlog_status keyword:

0 (zero)	Disables the binary event logger.
BINLOG_ON	Enables the binary event logger.
BINLOG_ASCIION	Enables the logging of kernel ASCII messages into the binary event log if the binary event logger is enabled. This value must be specified with the BINLOG_ON value as follows: `int binlog_status = BINLOG_ON | BINLOG_ASCII;`

After you modify the /sys/data/binlog_data.c file, you must rebuild and boot the new kernel.

14.4 Recovering Event Logs After a System Crash

You can recover unprocessed messages and binary event-log records from a system crash when you reboot the system.

The msgbuf.err entry in the /etc/syslog.conf file specifies the destination of the kernel syslog message buffer msgbuf that is recovered from the dump file. The default /etc/syslog.conf file entry for the kernel syslog message buffer file is as follows:

```
msgbuf.err              /var/adm/crash/msgbuf.savecore
```

The dumpfile entry in the /etc/binlog.conf file specifies the file name destination for the kernel binary event-log buffer that is recovered from the dump file. The default /etc/binlog.conf file entry for the kernel binary event-log buffer file is as follows:

```
dumpfile                /usr/adm/crash/binlogdumpfile
```

If a crash occurs, the syslogd and binlogd daemons cannot read the /dev/klog and /dev/kbinlog special files and process the messages and binary event records. When you reboot the system, the savecore command

runs and, if a dump file exists, recovers the kernel syslog message and binary event-log buffers from the dump file. After `savecore` runs, the `syslogd` and `binlogd` daemons are started.

The `syslogd` daemon reads the syslog message buffer file, checks that its data is valid, and then processes it in the same way that it normally processes data from the `/dev/klog` file, using the information in the `/etc/syslog.conf` file.

The `binlogd` daemon reads the binary event-log buffer file, checks that its data is valid, and then processes the file in the same way that it processes data from the `/dev/kbinlog` special file, using the information in the `/etc/binlog.conf` file.

After the `syslogd` and `binlogd` daemons are finished with the buffer files, the files are deleted.

14.5 Maintaining Log Files

If you specify full pathnames for the message destinations in the `/etc/syslog.conf` and `/etc/binlog.conf` files, the log files will grow in size. Also, if you configure the `syslogd` daemon to create daily directories and log files, eventually there will be many directories and files, although the files themselves will be small. Therefore, you must keep track of the size and the number of log files and daily directories and delete the files and directories if they become unwieldy.

You can also use the `cron` daemon to specify that log files be deleted. The following is an example of a `crontab` file entry:

```
5 1 * * * find /var/adm/syslog.dated -type d -mtime +5 -exec rm -rf '{}' \;
```

The previous command line causes all directories under `/var/adm/syslog.dated` that were modified more than 5 days ago to be deleted, along with their contents, every day at 1:05. Refer to the `crontab`(1) reference page for more information.

14.6 Environmental Monitoring

On any system, thermal levels can increase because of poor ventilation, overheating conditions, or fan failure. Without detection, an unscheduled shutdown could ensue causing the system's loss of data or damage to the system itself. By using Environmental Monitoring, the thermal state of AlphaServer systems can be detected and users can be alerted in time enough to recover or perform an orderly shutdown of the system.

This chapter discusses how Environmental Monitoring is implemented on AlphaServer systems.

14.6.1 Environmental Monitoring Framework

The Environmental Monitoring framework consists of four components: loadable kernel module and its associated APIs, Server System MIB subagent daemon, the `envmond` daemon, and the `envconfig` utility.

14.6.1.1 Loadable Kernel Module

The loadable kernel module and its associated APIs contain the parameters needed to monitor and return status on your system's threshold levels. The kernel module exports server management attributes as described in Section 14.6.1.1.1 through the kernel configuration manager (CFG) interface only. It works across all platforms that support server management, and provides compatibility for other server management systems under development. The kernel module is supported on all Alpha systems running Version 4.0A or higher of the Digital UNIX operating system.

The loadable kernel module does not include platform specific code (such as the location of status registers). It is transparent to the kernel module which options are supported by a platform. That is, the kernel module and platform are designed to return valid data if an option is supported, a fixed constant for unsupported options, or null.

14.6.1.1.1 Specifying Loadable Kernel Attributes

The loadable kernel module exports the parameters listed in Table 14–2 to the kernel configuration manager (CFG).

Table 14–2: Parameters Defined in the Kernel Module

Parameter	Purpose
`env_current_temp`	Specifies the current temperature of the system. If a system is configured with the KCRCM module, the temperature returned is in Celsius. If a system does not support temperature readings and a temperature threshold has not been exceeded, a value of -1 is returned. If a system does not support temperature readings and a temperature threshold is exceeded, a value of -2 is returned.
`env_high_temp_thresh`	Provides a system specific operating temperature threshold. The value returned is a hardcoded, platform specific temperature in Celsius.
`env_fan_status`	Specifies a noncritical fan status. The value returned is a bit value of zero (0). This value will differ when the hardware support is provided for this feature.

Table 14–2: Parameters Defined in the Kernel Module (cont.)

Parameter	Purpose
env_ps_status	Provides the status of the redundant power supply. On platforms that provide interrupts for redundant power supply failures, the corresponding error status bits are read to determine the return value. A value of 1 is returned on error; otherwise, a value of zero (0) is returned.
env_supported	Indicates whether or not the platform supports server management and environmental monitoring.

14.6.1.1.2 Obtaining Platform Specific Functions

The loadable kernel module must return environmental status based on the platform being queried. This section describes the kernel interfaces used. To obtain environmental status, the get_info() function is used. Calls to the get_info() function are filtered through the platform_callsw[] table.

The get_info() function obtains dynamic environmental data using the function types described in Table 14–3.

Table 14–3: get_info() Function Types

Function Type	Use of Function
GET_SYS_TEMP	Reads the system's internal temperature on platforms that have a KCRCM module configured.
GET_FAN_STATUS	Reads fan status from error registers.
GET_PS_STATUS	Reads redundant power supply status from error registers.

The get_info() function obtains static data using the HIGH_TEMP_THRESH function type, which reads the platform specific upper threshold operational temperature.

14.6.1.1.3 Server System MIB Subagent

The Server System MIB Agent, (which is an eSNMP sub-agent) is used to export a subset of the Environmental Monitoring parameters specified in the Server System MIB. The Digital Server System MIB exports a common set of hardware specific parameters across all server platforms on all operating systems offered by Digital. Table 14–4 maps the subset of Server System MIB variables that support Environmental Monitoring to the kernel parameters described in Section 14.6.1.1.1.

Table 14–4: Mapping of Server Subsystem Variables

Server System MIB Variable Name	Kernel Module Parameter
svrThSensorReading	env_current_temp
svrThSensorStatus	env_current_temp
svrThSensorHighThresh	env_high_temp_thresh
svrPowerSupplyStatus	env_ps_temp
svrFanStatus	env_fan_status

An SNMP MIB compiler and other tools are used to compile the MIB description into code for a skeletal subagent daemon. Communication between the subagent daemon and the eSNMP daemon is handled by interfaces in the eSnmp shared library (libesnmp.so). The subagent daemon must be started when the system boots and after the eSNMP daemon has started.

For each Server System MIB variable listed in Table 14–4, code is provided in the subagent daemon, which accesses the appropriate parameter from the kernel module through the CFG interface.

14.6.1.2 Monitoring Environmental Thresholds

To monitor the system environment, the envmond daemon is used. You can customize the daemon by using the envconfig utility. The following sections discuss the daemon and utility. For more information, see the envmond and envconfig reference pages.

14.6.1.2.1 Environmental Monitoring Daemon

By using the Environmental Monitoring daemon, envmond, threshold levels can be checked and corrective action can ensue before damage occurs to your system. Then envmond daemon performs the following:

- Queries the system for threshold levels.
- Broadcasts a message to users and provides corrective action when a high threshold level or redundant power supply failure has been encountered. When a fan failure is encountered, a message is broadcasted and an orderly shutdown ensues.
- Notifies users when a high temperature threshold condition has been resolved.
- Notifies all users that an orderly shutdown is in progress if recovery is not possible.

To query the system, the envmond daemon uses the base operating system command /usr/sbin/snmp_request to obtain the current values of the environment variables specified in the Server System MIB.

To enable Environmental Monitoring, the envmond daemon must be started during the system boot, but after the eSNMP and Server System MIB agents have been started. You can customize the envmond daemon using the envconfig utility.

14.6.1.2.2 Customizing the envmond Daemon

You can use the envconfig utility to customize how the environment is queried by the envmond daemon. These customizations are stored in the /etc/rc.config file, which is read by the envmond daemon during startup. Use the envconfig utility to perform the following:

- Turn environmental monitoring on or off during the system boot.
- Start or stop the envmond daemon after the system boot.
- Specify the frequency between queries of the system by the envmond daemon.
- Set the highest threshold level that can be encountered before a temperature event is signaled by the envmond daemon. Specify the path of a user defined script that you want the envmond daemon to execute when a high threshold level is encountered.
- Specify the grace period allotted to save data if a shutdown message has been broadcasted.
- Display the values of the Environmental Monitoring variables.

A

Device Mnemonics

This appendix identifies and defines the mnemonics that you use to attach any hardware or software device to your system. You specify the mnemonics with the MAKEDEV command to create the character or block special files that represent each of the devices. You also use the mnemonics to specify device special files for the loadable drivers described in the /etc/sysconfigtab configuration database file.

Table A–1 lists the mnemonics in six categories: generic, consoles, disks, tapes, terminals, and printers. The generic category lists the mnemonics of a general nature and includes memory, null, trace, and tty devices. The consoles category lists mnemonics for the system console devices that the Digital UNIX operating system uses. The disks, tapes, terminals, and printers categories identify the appropriate mnemonics for those devices.

The Description column in Table A–1 identifies the corresponding device name. It does not define the mnemonic's use. For detailed information on the use of each mnemonic in relation to the MAKEDEV command, the cfgmgr configuration manager daemon, and the system configuration file, use the man command. For example, enter the following command to display the reference page for the SCSI disk controller driver:

```
% man rz
```

Where appropriate, the reference page defines the device's syntax as it should appear in the config file. For additional software device mnemonics that the MAKEDEV command uses, refer to the MAKEDEV(8) reference page.

Note

Table A–1 uses the convention of an * (asterisk) beside a mnemonic and a ? (question mark) beside a device name to indicate a variable number. The value of the variable number is dependent on the particular device.

Table A–1: Device Mnemonics

Category	Mnemonic	Description
Generic	std	Standard devices with all console subsystems
	drum	Kernel drum device
	kmem	Virtual main memory
	mem	Physical memory
	null	A null device
	trace	A trace device
	tty	A tty device
	local	Customer-specific devices
Prestoserve	nvtc	DEC 3000 Model 300, DEC 3000 Model 400, DEC 3000 Model 500, DEC 3000 Model 600, DEC 3000 Model 800
Consoles	console	System console interface
Disks	rz*	SCSI disks (RZ24L/RZ25/RZ25L/RZ26/RZ26L/RZ28/RZ28B/RZ55/ RZ56/RZ58/RZ73/RZ74/RX23/RX26/RX33/RRD42/HSZ10)
	ra*	DSA disks (RA60/RA70/RA71/RA72/RA73/RA80/RA81/RA82/ RA90/RA92)
Tapes	tz*	SCSI tapes (TKZ08/TKZ09/TLZ04/TLZ06/TLZ07/TSZ07/TZ30/TZ85/ TZK10/TZK11)
	ta*	DSA tapes (TA78/TA79/TA90/TA91)
Terminals	pty	Network pseudoterminals
Modems		DF02/DF03/DF296

Table A–1: Device Mnemonics (cont.)

Category	Mnemonic	Description
Printers		LA50, LA70, LA75, LA324, LA424, LF01R, LG02, LG06, LG12, LG31, LJ250, LN03, LN03S, LN03R, LN05, LN05R, LN06, LN06R, LN07, LN07R, LN08, LN08R, IBMPRO, NEC290, FX80, FX1050, HPIIP, HPIIIP, HPIIID, HPIV, HP4M

B

SCSI/CAM Utility Program

B.1 Introduction

The SCSI/CAM Utility Program, scu, interfaces with the Common Access Method (CAM) I/O subsystem and the peripheral devices attached to Small Computer System Interface (SCSI) busses. This utility implements the SCSI commands necessary for normal maintenance and diagnostics of SCSI peripheral devices and the CAM I/O subsystem.

The format of a SCU command is as follows:

scu> [-f *device-name-path*] [*command* [*keyword*]...]

If a device name is not specified in one of the *options* on the command line, the program checks the SCU_DEVICE environment variable to determine the device name. If SCU_DEVICE is not set, you must use the `set nexus` command to select the device and operation or some commands may be restricted.

For example, if you do not specify a device name and SCU_DEVICE is not set, you cannot format a disk because the scu utility cannot perform a mounted file system check. See Section B.3 for a description of the `set` command and its arguments.

If a command is not entered on the command line, the program prompts for commands until you terminate the program. In most cases, you can abbreviate commands to the lowest unambiguous number of characters.

This appendix contains an overview of the scu functions that system administrators use. Detailed information, including examples of use, is available through the online help for the scu utility. To use the help facility once you are in the scu utility, issue the `help` command at the scu> prompt.

B.2 SCU Utility Conventions

The following conventions describe the scu utility syntax:

Convention	Meaning
keyword (*alias*)	Use a keyword or the specified alias.
address-format	Optionally accepts an address format.
nexus-information	Optionally accepts nexus information.
test-parameters	Optionally accepts test parameters.
D: *value* or *string*	The value or string shown is the default.
R: *minimum-maximum*	Enter a value within the range specified.

The *address-format* parameter is optional. It is available for use with most CD–ROM Show Audio commands that specify the address format of information returned by the drive. The possible address formats are as follows:

Format	Description
lba	Logical block address
msf	Minute, second, and frame

The syntax of a command that uses the *address-format* parameter is as follows:

scu> [*command*] [*address-format*{lba|msf}]

The *nexus-information* parameter lets users specify values to override the bus, target, and LUN values normally taken from the selected SCSI device. The *nexus-information* keywords are as follows:

Parameter	Description
bus (pid) R:0-3	SCSI bus number (path ID)
target (tid) R:0-7	SCSI target number (target ID)
lun R:0-7	SCSI Logical Unit Number (LUN)

You use the *test-parameter* variables to specify the physical limits of the media on which the command can operate. For example, these may be the starting and ending logical block numbers on a disk. The test parameters for a command use the following syntax:

scu> *command* [*media-limits*] [*test-control*]

The *media-limits* parameter, which controls the media tested, has the following syntax:

```
                                    { lb n }   { length n }
scu> command [{ starting n }] [{ endingn }] [size n]
                                    { limit  n }
                                    { records n }
```

The alias `bs` (block size) is accepted for the *`size`* keyword.

The *`test-control`* parameters control aspects of the test operation. The *`test-control`* parameters supported are:

```
                                    { align Align-Offset }
                                    { compare { on | off } }
scu> command [ { errors Error-Limit  } ]
                                    { passes Pass-Limit    }
                                    { pattern Data-Pattern  }
                                    { recovery { on | off } }
```

B.3 General SCU Commands

This section describes the general-purpose `scu` utility commands. For more information on each command, see the online help that is part of the `scu` utility.

evaluate *`expression`*

This command evaluates the given expression and displays values in decimal, hexadecimal, blocks, kilobytes, megabytes, and gigabytes. The expression argument is the same as that described for test parameter values. The output depends on whether the verbose display flag is set.

The following examples show the output of the `evaluate` command. Verbose mode is turned on for the first two `evaluate` commands and turned off for the last one.

```
scu>set verbose on
scu>evaluate 0xffff
Expression Values:
            Decimal: 65535
        Hexadecimal: 0xffff
    512 byte Blocks: 128.00
          Kilobytes: 64.00
          Megabytes: 0.06
          Gigabytes: 0.00
scu>evaluate 64k*512
Expression Values:
             Decimal: 33554432
         Hexadecimal: 0x2000000
     512 byte Blocks: 65536.00
           Kilobytes: 32768.00
           Megabytes: 32.00
           Gigabytes: 0.03
scu>set verbose off
scu>evaluate 0xffff
Dec: 65525 Hex: 0xffff Blks: 128.00 Kb: 64.00 Mb: 0.06 Gb: 0.00
```

exit

You use this command to exit from the program. You can use `quit` as an alias for `exit`. You can terminate the program in interactive mode by entering the end-of-file character (usually Ctrl/d).

help [*topic*]

This command displays help information on topics. You can use a question mark (?) as an alias. If you issue the `help` command without specifying a topic, a list of all available topics is displayed.

scan [*edt, nexus-information, report-format*]
scan [*media test-parameters*]

This command scans either device media or the CAM Equipment Device Table (EDT).

The following examples use the `scan edt` command. The first example illustrates the command followed by the `show device` command to display the information resulting from the scan.

```
scu> scan edt
Scanning bus 1, target 6, lun 0, please be patient...
scu> show device
Scanning bus 1, target 6, lun 0, please be patient...
Inquiry Information:
                     SCSI Bus ID: 1
                  SCSI Target ID: 6
                 SCSI Target LUN: 0
           Peripheral Device Type: Direct Access
             Peripheral Qualifier: Peripheral Device Connected
            Device Type Qualifier: 0
                  Removable Media: No
                     ANSI Version: SCSI-1 Compliant
                     ECMA Version: 0
                      ISO Version: 0
            Response Data Format: CCS
               Additional Length: 31
           Vendor Identification: DEC
          Product Identification: RZ55     (C) DEC
        Firmware Revision Level: 0700
scu> scan edt bus 1
Scanning bus 1, target 6, lun 0, please be patient...
```

The `media` argument causes the device media to be scanned. This involves writing a data pattern to the media and then reading and verifying the data written. You must include test parameters that specify the media area to be scanned.

The following examples use the `scan media` command with different *test-parameters*:

```
scu> media
scu: No defaults, please specify test parameters for transfer
scu> scan media length 100 recovery off
Scanning 100 blocks on /dev/rrz10c (RX23) with pattern
                                        0x39c39c39...
```

```
scu> scan media lba 200 limit 25k align 'lp-1'
Scanning 50 blocks on /dev/rrz10c (RX23) with pattern
                                        0x39c39c39...
scu> scan media starting 0 bs 32k records 10
Scanning 640 blocks on /dev/rrz10c (RX23) with pattern
                                         0x39c39c39...
Scanning blocks [ 0 through 63 ]...
Scanning blocks [ 64 through 127 ]...
Scanning blocks [ 128 through 191 ]...
Scanning blocks [ 192 through 255 ]...
Scanning blocks [ 256 through 319 ]...
Scanning blocks [ 320 through 383 ]...
Scanning blocks [ 384 through 447 ]...
Scanning blocks [ 448 through 511 ]...
Scanning blocks [ 512 through 575 ]...
Scanning blocks [ 576 through 639 ]...
```

set

The set command sets parameters for a device or sets environment parameters for the scu program. See the on-line help text that is part of the scu utility for an explanation of each parameter.

```
        { audio keywords }
        { cam debug hex-flags }
        { debug { on | off } }
        { default parameter }
  set   { device device-type }
        { dump { on | off } }
        { dump-limit value }
        { log file-name-path }
        { nexus nexus-information }
        { pages [ mode-page [ pcf page-control-field ] }
        { pager paging-filter }
        { paging { on | off } }
        { recovery { on | off } }
        { tape keywords ... }
        { verbose { on | off } }
        { watch { on | off } }
```

show

You use this command to display parameters for a device or the program. See the scu online help for more information.

```
        { audio keywords }
        { capacity }
        { defects }
        { device }
show    { edt }
        { memory }
        { mode-pages }
        { nexus }
        { pages }
        { path-inquiry }
```

source*input-file*
: This command allows you to source input from an external command file. If any errors occur during command parsing or execution, the command file is closed at that point. The default file name extension `.scu` is appended to the name of the input file if no extension is supplied. If the `scu` utility cannot find a file with the `.scu` extension, it attempts to locate the original input file.

switch [*device-name*]
: This command accesses a new device or a previous device. If no device name is specified, the command acts as a toggle and simply switches to the previous device, if one exists. If a device is specified, it is validated and becomes the active device.

B.4 Device and Bus Management Commands

This section describes the following `scu` utility commands for managing SCSI devices and the CAM I/O subsystem:

allow
: This command allows media to be removed from the selected device.

eject
: You use this command with CD–ROMs to stop play and eject the caddy.

mt *command* [*count*]
: This command issues one of the supported `mt` commands. See the online help text that is part of the `scu` utility for information on the `mt` commands.

pause
: You use this command to pause the playing of a CD–ROM audio disc.

play
: You use this command to play audio tracks on a CD–ROM audio disc. If no keywords are specified, all audio tracks are played by default. You can specify a track number, a range of audio tracks, a logical block address, or a time address. See the online help that is part of the `scu` utility for information on the `play` command.

prevent
: This command prevents media removal from the selected device.

release {*device* | simqueue} [*nexus-information*]

This command releases a reserved SCSI device or releases a frozen SIM queue after an error. The *device* argument specifies a reserved SCSI device to be released. The extent release capability for direct access devices is not implemented.

The `simqueue` argument issues a release SIMQ CCB to thaw a frozen SIM queue. Ordinarily, this command is not necessary because the SIM queue is automatically released after errors occur. If the nexus information is omitted, the SIM queue for the selected SCSI device is released.

The following example shows the `release` command:

```
scu> release simqueue bus 1 target 6 lun 0
```

reserve *device*

This command issues a SCSI Reserve command to the selected device. The entire logical unit is reserved for the exclusive use of the initiator. Extent reservation for direct access devices is not implemented.

reset {bus | device} [*nexus-information*]

This command resets the SCSI bus or the selected SCSI device.

The `bus` argument issues a CAM Bus Reset CCB. If the nexus information is omitted, the bus associated with the selected SCSI device is reset. The `reset bus` command is restricted to superuser (root) access because it can cause loss of data to some devices.

The `device` argument issues a CAM Bus Device Reset CCB. If the nexus information is omitted, the selected device is reset. The `reset device` command requires write access to the selected device because the command can cause loss of data to some devices.

If nexus information is specified, this command is restricted to the superuser.

`resume`

This command causes a CD–ROM audio disc to resume play after it has been paused with the `pause` command.

start

This command issues a SCSI Start Unit command to the selected device. This action enables the selected device to allow media access operations.

stop

This command issues a SCSI Stop Unit command to the selected device. This action disables the selected device from allowing media access operations.

tur

This command issues a Test Unit Ready command to determine the readiness of a device. If the command detects a failure, it automatically reports the sense data.

verify media [*test-parameters*]

This command performs verify operations on the selected device.

The `media` argument verifies the data written on the device media. This activity involves reading and performing an ECC check of the data. If the test parameters are omitted, the entire device media is verified.

If the device does not support the `verify` command, the following error message appears:

```
scu>verify media starting 1000 length 1024
Verifying 1024 blocks on /dev/rrz10c (RX23),
                                please be patient...
Verifying blocks [ 1000 through 2023 ] ...
scu: Sense Key = 0x5 = ILLEGAL REQUEST -
                                Illegal request or CDB parameter,
     Sense Code/Qualifier = (0x20, 0) =
                                Invalid command operation code
```

When an error occurs, the sense key is examined. The expected sense keys are Recovered Error (0x01) or Medium Error (0x03). When these errors are detected, the following error message is displayed and verification continues with the block following the failing block:

```
scu: Verify error at logical block number 464392 (0x71608).
scu: Sense Key = 0x1 = RECOVERED ERROR -
                            Recovery action performed,
     Sense Code/Qualifier = (0x17, 0) = Recovered data with no
                            error correction applied
```

If any other sense key error occurs, the full sense data is displayed and the verification process is aborted.

The following conditions apply to the `verify` command:

- On failure, the failing logical block number (LBN) is reported and verification continues with the block following the failing block.
- By default, verification is performed using the current parameters in the Error Recovery mode page. You can disable drive recovery by using the `set recovery off` command.

For example:

```
scu> verify media lba 464388
Verifying 1 blocks on /dev/rrz14c (RZ55),
                          please be patient...
Verifying blocks [ 464388 through 464388 ] ...
scu> verify media starting 640000
Verifying 9040 blocks on /dev/rrz14c (RZ55),
                             please be patient...
Verifying blocks [ 640000 through 649039 ] ...
scu> verify media starting 1000 length 250
Verifying 250 blocks on /dev/rrz14c (RZ55),
                            please be patient...
Verifying blocks [ 1000 through 1249 ] ...
scu> verify media starting 1000 ending 2000
Verifying 1001 blocks on /dev/rrz14c (RZ55),
                             please be patient...
Verifying blocks [ 1000 through 2000 ] ...
```

B.5 Device and Bus Maintenance Commands

This section describes scu utility commands for maintaining SCSI devices and the CAM I/O subsystem.

The following command changes the mode pages for a device:

change pages [*mode-pages...*] [pcf *page-control-field*]

The program prompts you with a list of the page fields that are marked as changeable. If you do not specify a mode page, all pages supported by the device are requested for changing. After you enter the new fields for each page, you use a mode select command to set the new page parameters.

The *mode-page* argument describes the mode page to change. The mode pages are as follows:

scu Keyword	Page Code	Description
error-recovery	0x01	Error recovery page
disconnect	0x02	Disconnect/reconnect page
direct-access	0x03	Direct access format page
geometry	0x04	Disk geometry page
flexible	0x05	Flexible disk page
cache-control	0x08	Cache control page
cdrom	0x0D	CD–ROM device page
audio-control	0x0E	Audio control page
device-configuration	0x10	Device configuration page
medium-partition-1	0x11	Medium partition page 1

scu Keyword	Page Code	Description
dec-specific	0x25	Digital-specific page
readahead-control	0x38	Read-ahead control page

Notes on the change pages command:

- Only fields that are marked changeable in the changeable mode page are prompted for.
- The default page control field is "current" values.
- Selecting a "pcf" is sticky (for example, sets new "pcf" default).
- Changing page values always affects the current page values.
- The default is to save page values if the page saveable bit in the page header is set and the program savable flag is set. Use the set default savable command to alter this flag.
- Some pages, such as those that affect the physical media, do not actually get saved until the media is formatted.

For mode pages that are unknown to the program, you can specify a hex page code for the page to change. In this mode, mode page fields are displayed and changed by hex byte values. You can also use this format to override the known formatted page change functions. For example:

change page code *hex-code* [pcf [*page-control-field*]]

The following example shows the change page command with the code parameter:

```
scu> change page code 0x21
Changing Unknown Page Parameters (Page 21 - current values):
Byte 2 [R:0-0xff D:0x2]: 3
Byte 3 [R:0-0xff D:0x8]:
Byte 4 [R:0-0xff D:0]:
Byte 5 [R:0-0xff D:0]:
Byte 6 [R:0-0xff D:0x3]:
Byte 7 [R:0-0xff D:0xe8]:
Byte 8 [R:0-0xff D:0x2]:
Byte 9 [R:0-0xff D:0x96]:
Byte 10 [R:0-0xff D:0x5]:
scu>
```

The *page-control-field* argument specifies the type of mode pages to obtain from the device. The page control fields that you can specify are as follows:

- changeable

- current
- default
- saved

The following example changes the error recovery parameters:

```
scu> change pages error
Changing Error Recovery Parameters (Page 1 - current values):
Disable Correction (DCR) [R:0-1 D:0]:
Disable Transfer on Error (DTE) [R:0-1 D:0]:
Post Recoverable Error (PER) [R:0-1 D:0]:
Transfer Block (TB) [R:0-1 D:0]:
Automatic Write Allocation (AWRE) [R:0-1 D:1]:
Read Retry Count [R:0-255 D:8]: 25
Write Retry Count [R:0-255 D:2]: 5
scu>
```

download *filename* [save] [{id *buffer-id*|offset *offset-value*|segment[*size*]}]

You can use the preceding command with any device that supports the downloading of operating software through the Write Buffer command.

The save keyword directs the device to save the new operating software in nonvolatile memory if the download command is completed successfully. With save specified, the downloaded code remains in effect after each power cycle and reset. If the save keyword is not specified, the downloaded software is placed in the control memory of the device. After a power cycle or reset, the device operation would revert to a vendor-specific condition.

If the save parameter is omitted, a Download microcode (mode 4) command is issued. Specifying save performs a Download microcode and save operation (mode 5). Not all devices accept both modes.

You can specify various parameters to control the download operation. Most devices do not require these optional parameters, but since each vendor may implement the download command differently, these parameters provide the capability to override program defaults.

The following notes apply to the download parameters:

- The default buffer ID is 0.
- The default offset value is 0.
- The default segment size is 8 KB. If this parameter is not specified, the default is to download the entire image at once.
- Refer to the vendor's SCSI programming manual for information on buffer ID's and buffer modes supported.

The following notes apply to the download command:

- If you enter scu using the default device /dev/cam and then set the device to download using the set nexus command, the code associated with checking for mounted file systems will fail. This was done purposely to prevent accidental downloading of disks with mounted file systems.
- Some devices, such as many disks, require additional time after the download operation to program the flash memory (save the firmware) to recalibrate or perform other necessary setup before the device can be accessed. In most cases, waiting 1 to 3 minutes is advised before accessing the device. Most devices will not respond to a selection immediately after a download operation.
- If a disk device determines that a recalibration is necessary, the drive may be unavailable for up to 10 minutes. During this sequence, you can usually issue the Test Unit Ready (tur) command to determine if the calibration has completed. If the calibration sequence is interrupted, for example by a bus reset, device reset, or by power cycling, the recalibration will be restarted when the drive is powered up.
- Do not power cycle devices during the download operation. Doing so may render your drive useless.

The following examples show the download command:

```
scu> download ???.fup save
Downloading & Saving Firmware File '???.fup' of 131076 bytes...
scu> download ???.fup save segment
Downloading File '???.fup' of 131076 bytes in 8192 byte
                                                segments...
Download completed successfully, now saving the microcode...
scu> download ???.fup save segment 32k
Downloading File '???.fup' of 131076 bytes in 32768 byte
                                                  segments...
Download completed successfully, now saving the microcode...
```

format [density *density-type*] [defects *defect-list*]

This command formats both hard and flexible disk media. Since this command modifies the disk media, the full command name must be entered to be recognized.

The *density-type* parameter specifies the density type for flexible disk media.

The *defect-list* parameter can be all, primary, or none. The default is to format with all known defects. If you use the default device /dev/cam to enter the scu utility and then use the set nexus command to set the device to format, the code associated with checking for mounted file systems fails. This failure avoids the possibility of accidentally formatting disks with mounted file systems.

read media [*test-parameters*]

This command performs read operations from the selected device. The command reads the device media and performs a data comparison of the data read. You must include test parameters that specify the media area to be read.

The examples that follow illustrate the use of the `read` command with several test-parameters:

```
scu> read media
scu: No defaults, please specify test parameters for
transfer...
scu> read media lba 100
Reading 1 block on /dev/rrz10c (RX23) using pattern
0x39c39c39...
scu> read media lba 100 pattern 0x12345678
Reading 1 block on /dev/rrz10c (RX23) using pattern
0x12345678...
scu: Data compare error at byte position 0
scu: Data expected = 0x78, data found = 0x39
scu> read media ending 100 compare off bs 10k
Reading 101 blocks on /dev/rrz10c (RX23)...
Reading blocks [ 0 through 19 ]...
Reading blocks [ 20 through 39 ]...
Reading blocks [ 40 through 59 ]...
Reading blocks [ 60 through 79 ]...
Reading blocks [ 80 through 99 ]...
```

reassign lba *logical-block*

This command allows you to reassign a defective block on a disk device. Since this command modifies the disk media, the full command name must be entered to be recognized.

test [controller | drive | memory | selftest]

This command performs tests on a controller by issuing send and receive diagnostic commands or `write buffer` and `read buffer` commands for memory testing to the selected device. If you issue the `test` command with no arguments, the utility performs a self test, which is supported by most controllers.

test memory [*test-parameters*] [id *buffer-id* | mode *buffer-mode* | offset *offset-value*]

This command verifies the controller memory by using the SCSI Read and Write Buffer commands. Since most controllers accept only the Combined Header and Data Mode with a buffer ID and buffer offset of zero, these are the defaults, but may be overridden. By default, the full memory size returned in the Read Buffer header is written/read/verified, but this too may be overridden by specifying a smaller data limit or size.

You can use various parameters to control the test memory operation. Most devices do not require these optional parameters, but newer devices may require different parameters to access the controller data buffer.

The following notes apply to the parameters used with the `test memory` command:

- The default buffer ID is restricted to 0, since this ID normally selects the controller data buffer.
- The default buffer mode is 0 (selects Combined Header and Data Mode).
- The default offset value is 0. The program automatically adjusts the memory data bytes being tested when this parameter is nonzero.
- Because writing to the controller data buffer destroys data that may be valid for active I/O requests, do not use this command for disk devices with mounted file systems.
- Refer to the vendor's SCSI programming manual for information on buffer IDs and buffer modes supported.

If the device does not support a Read Buffer command and/or the default parameters, an error message similar to the following is displayed:

```
scu> test memory
      Performing Controller Memory Diagnostics...
      Testing Controller Memory of 245760 bytes (Mode 0,
Offset 0)
      Testing 245760 bytes on [1/2/0] (TZK11) using pattern
0x39c39c39...
      scu: 'SCMD_WRITE_BUFFER' failed, EIO (5) - I/O error
      scu: Sense Key = 0x5 = ILLEGAL REQUEST -
                                  Illegal request or CDB
parameter,
           Sense Code/Qualifier = (0x24, 0) =
                                    Invalid field in cdb
```

The following examples show the `test memory` command:

```
% scu -f /dev/rrz11c
scu> test memory
Performing Controller Memory Diagnostics...
      Testing Controller Memory of 61376 bytes (Mode 0,
Offset 0)
      Testing 61376 bytes on /dev/rrz11c (RZ56) using pattern
0x39c39c39...
scu> test memory pattern 0x12345678 size 50k
      Performing Controller Memory Diagnostics...
      Testing Controller Memory of 61376 bytes (Mode 0,
Offset 0)
      Testing 51200 bytes on /dev/rrz11c (RZ56) using pattern
0x12345678...scu> test memory passes 5
 Performing Controller Memory Diagnostics...
      Testing Controller Memory of 61376 bytes (Mode 0,
Offset 0)
      Testing 61376 bytes on /dev/rrz11c (RZ56) using pattern
0x39c39c39...
      Testing 61376 bytes on /dev/rrz11c (RZ56) using pattern
```

```
0x00ff00ff...
      Testing 61376 bytes on /dev/rrz11c (RZ56) using pattern
0x0f0f0f0f...
      Testing 61376 bytes on /dev/rrz11c (RZ56) using pattern
0xc6dec6de...
      Testing 61376 bytes on /dev/rrz11c (RZ56) using pattern
0x6db6db6d...
scu> show memory
The Controller Memory Size is 245760 (0x3c000) bytes.
scu> test memory mode 2
Performing Controller Memory Diagnostics...
      Testing Controller Memory of 245760 bytes (Mode 2,
Offset 0)
      Testing 245760 bytes on [1/2/0] (TZK11) using pattern
0x39c39c39...
```

write [media *test-parameters*]

The media argument writes to the device media by using various data patterns. The patterns default to 0x39c39c39 for the first pass, 0xc6dec6de for the second, and so on as shown in the last example. You must specify transfer parameters that specify the media area to be written.

The following examples show the write media command:

```
scu> write media
 No defaults, please specify test parameters for
transfer...

scu> write media lba 100
Writing 1 block on /dev/rrz10c (RX23) with pattern
0x39c39c39...
scu> write media starting 100 ending 250
Writing 151 blocks on /dev/rrz10c (RX23) with pattern
0x39c39c39...
scu> write media starting 2800 limit 1m bs 10k
Writing 80 blocks on /dev/rrz10c (RX23) with pattern
0x39c39c39...
Writing blocks [ 2800 through 2819 ]...
Writing blocks [ 2820 through 2839 ]...
Writing blocks [ 2840 through 2859 ]...
Writing blocks [ 2860 through 2879 ]...
scu> write media lba 2879 passes 5
Writing 1 block on /dev/rrz10c (RX23) with pattern
0x39c39c39...
Writing 1 block on /dev/rrz10c (RX23) with pattern
0xc6dec6de...
Writing 1 block on /dev/rrz10c (RX23) with pattern
0x6db6db6d...
Writing 1 block on /dev/rrz10c (RX23) with pattern
0x00000000...
Writing 1 block on /dev/rrz10c (RX23) with pattern
0xffffffff...
```

C

Support of the CI and HSC Hardware

The Computer Interconnect (CI) bus is a high-speed, dual-path bus that connects processors and Hierarchical Storage Controllers (HSCs) in a computer room environment. An HSC is an I/O subsystem that is a self-contained, intelligent mass storage controller that provides access to disks and tapes from multiple host nodes attached to the CI bus.

Note

The Digital UNIX implementation has the following limitations:

- You can attach a maximum of four HSCs to a CI bus.
- You can attach a single CI bus to a host.
- Under no circumstances can a Digital UNIX node participate as a VMS cluster member. A configuration that includes a VMS system and a Digital UNIX system residing on the same CI bus is not supported.

Digital UNIX supports Digital's System Communication Architecture (SCA) for CI port adapters and HSCs. SCA implements port and class driver support, and standardizes the ways in which TMSCP (tms) and MSCP (ra) devices are handled. SCA separates features into different architectural layers, thus minimizing the effect that software changes to one layer have on other layers.

C.1 Hardware Setup, Restrictions, and Revision Levels

For information on physical components and setup, refer to the HSC hardware documentation and the hardware documentation for your processor and supported devices. Only processors with CI adapters can support HSC configurations.

When setting up the HSC controller hardware, you should attach a terminal to the HSC in order to use commands to get or set HSC parameters, to monitor connections between remote systems, and to identify the disk or tape status.

The maximum number of hosts on a CI bus is 16. The host number for any host on the CI bus must be between 0 and 15.

Note

Two parameters of particular importance are the system ID and the system name. Do not duplicate any system identification or names of nodes on the star coupler.

C.2 Software Installation and Restrictions

The installation software assists you in identifying and configuring the components of your system. You should be familiar with the basic installation guide for your processor before starting the actual installation.

During installation of the Digital UNIX software, each accessible MSCP (ra) disk device must be uniquely identified by its unit plug number as follows:

- The unit plug number must be between 0 and 254, inclusive.
- Each unit plug number must be unique. Two different disks cannot have the same unit plug number even if the disks are on separate controllers. For example, if the unit plug number for a disk on controller A is 5 and the unit plug number for a separate disk on controller B is also 5, you must change one of the numbers.
- You can connect a disk with a unique unit plug number to two different controllers (dual or porting). Refer to the `ra`(7) reference page for information on how to specify the device entry in the system configuration file.

C.3 Configuration File Entries

The installation software ensures that your HSC components are configured into the kernel and are included in the `/usr/sys/conf/`*`NAME`* system configuration file, where *`NAME`* specifies your system name in uppercase letters.

Chapter 5 provides information on the following entries that correspond to a CI or HSC configuration:

- Description of the `scs_sysid` parameter
- CI adapter specifications
- Controller and device specifications

C.4 Booting an HSC Controller or an HSC Disk

The Digital UNIX software supports booting an HSC disk on the DEC 7000 and DEC 10000 processors. If an HSC controller fails, any disks connected to that HSC controller are inaccessible. Attempts to access those disks will cause the accessing system to hang until the HSC reboots completely. Refer to your processor hardware documentation for explicit instructions on booting an HSC disk.

C.5 Sharing Disk and Tape Units Among Several Hosts

Although an HSC can be shared among several hosts, there is no software interlocking mechanism to prevent concurrent write operations to the same partition by multiple Digital UNIX systems. The following restrictions must be observed:

- Only multiple readers can share a disk unit; writable file systems cannot be shared.
- If a disk will be shared, it should be hardware write protected.
- Each host must mount the file systems to be accessed with the read-only (`-r`) option to the `mount` command.
- Only a single host can mount a disk that contains writable file systems.
- Use the Network File System (NFS) if multiple writers need to share partitions.

You should coordinate disk unit ownership among the hosts on the CI bus, for example, assign a range of disk unit numbers to each host. The HSC controller can also be directed to limit disk access to an exclusive host system. This limitation protects the disk from accidental access by another host on the CI bus. For more information, see the `radisk`(8) reference page, in particular the `–e` and `–n` options.

Tape drives that are attached to an HSC controller can be shared. This feature is recommended and provides greater use of tape drives. Be aware that the access mechanism provides serial sharing of the drives, not simultaneous access.

D

Using the uerf Event Logger

Note

Information about the uerf event logger is provided in this appendix for backward compatibility reasons. The uerf component will be retired in a future release of the Digital UNIX operating system. The replacement event logger is the DECevent utility. For more information about DECevent, see Chapter 14 and dia(8).

Use the uerf command to produce event reports from the binary log file. You must be superuser to use the uerf command. The uerf command accesses events logged to the binary log file, translates them from binary code to ASCII if necessary, and sends them to the output device you specify. The events include error messages relating to the system hardware and the software kernel, as well as information about system status, startup, and diagnostics. The default binary log file is /usr/adm/binary.errlog.

By reviewing the types and the number of events, you can determine the reliability of a system. If a report shows a large number of errors for a particular device, you can determine if a problem exists before the device fails completely. Furthermore, if a failure occurs, the event report provides information on the events that led to the failure.

The uerf command uses the following three data files:

- /usr/sbin/uerf.bin – The event information database
- /usr/sbin/uerf.hlp – The help file
- /usr/sbin/uerf.err – The error message file

The uerf command allows you to specify the source that it uses to generate event reports, to restrict the event selection, and to produce specific output formats. The uerf command has the following syntax:

/usr/sbin/uerf [*options*...]

Without options, the uerf command outputs the contents of the event-log file specified by the *.* entry in the /etc/binlog.conf configuration file.

To report on any other event-log file or if there is no *.* entry, you must use the `uerf` command with the `–f` option.

Table D–1 describes the `uerf` command options.

Table D–1: Options to the uerf Command

Option	Description
`–c` *`class,...`*	Selects events for the specified classes.
`–D` [*`disk,...`*]	Selects events for the specified mscp and SCSI disk devices.
`–f` *`filename`*	Specifies the file from which messages are read.
`–h`	Displays help information.
`–H` *`hostname`*	Selects events for the specified host system. This option is used if events from multiple systems are being forwarded to the local host.
`–M` [*`mainframe,...`*]	Specifies processor event types.
`–n`	Processes events as they occur.
`–o` *`output`*	Produces output in either `brief`, `full`, or `terse` format. The default is `brief`.
`–O` [*`op_events,...`*]	Selects the specified operating system events.
`–R`	Produces output in reverse chronological order.
`–r` *`record,...`*	Selects events for the specified record codes.
`–s` [*`seq_of_numbers`*]	Selects events with the specified sequence of numbers.
`–S`	Produces a summary report.
`–t` *`time`*	Selects events within the specified time range.
`–T` [*`tapes,...`*]	Selects events for TMSCP tape types and SCSI tape devices.
`–u` *`number`*	Selects events from the device with the specified unit number.
`–x`	Excludes specified options.
`–Z`	Produces output in hexadecimal format.

To use the `uerf` command in single-user mode, you must ensure that the file system containing the binary log file and the `uerf` command data files is mounted.

The line printer spooler is not operational during single-user mode. Therefore, to print a report on a line printer while in single-user mode, you

must direct the output to a printer special file as shown in the following example:

```
# /usr/sbin/uerf > /dev/lp
```

You can use some options together. For example, the following command produces a report from the /var/admin/logs.old file for the system guitar:

```
# /usr/sbin/uerf –f /var/admin/logs.old –H guitar
```

The following example uses the –t and the –o options to display messages for the current day in terse format:

```
# /usr/sbin/uerf -t s:00 -o terse
```

The following example shows the default output of the uerf command:

```
# /usr/sbin/uerf
     uerf version 4.2-011 (118)

************************ ENTRY  1. **************************

----- EVENT INFORMATION -----

EVENT CLASS                     OPERATIONAL EVENT
OS EVENT TYPE           300.    SYSTEM STARTUP
SEQUENCE NUMBER           0.
OPERATING SYSTEM                DEC OSF/1
OCCURRED/LOGGED ON              Tue Jan 11 17:16:18 1994
OCCURRED ON SYSTEM              pearl
SYSTEM ID          x0004000F    CPU TYPE: DEC
                                CPU SUBTYPE: KN15AA
MESSAGE                         Alpha boot: available memory from
                                 _0x646000 to 0x6000000
                                DEC OSF/1 X1.2-11  (Rev. 4); Tue Jan
                                 _11 17:13:53 EST 1994
                                physical memory = 94.00 megabytes.
                                available memory = 85.48 megabytes.
                                using 360 buffers containing 2.81
                                _megabytes of memory
                                tc0 at nexus
                                scc0 at tc0 slot 7
                                asc0 at tc0 slot 6
                                rz1 at asc0 bus 0 target 1 lun 0 (DEC
                                 _   RZ25     (C) DEC 0700)
                                rz2 at asc0 bus 0 target 2 lun 0 (DEC
                                 _   RZ25     (C) DEC 0700)
                                tz5 at asc0 bus 0 target 5 lun 0 (DEC
                                 _   TLZ06     (C)DEC 0374)
                                asc1 at tc0 slot 6
                                fb0 at tc0 slot 8
                                1280X1024
                                ln0: DEC LANCE Module Name: PMAD-BA
                                ln0 at tc0 slot 7
                                ln0: DEC LANCE Ethernet Interface,
                                 _hardware address: 08:00:2b:2c:f6:9f
                                DEC3000 - M500 system
                                Firmware revision: 2.0
                                PALcode: OSF version 1.28
                                lvm0: configured.
```

```
                              lvm1: configured.
                              setconf: bootdevice_parser translated
                               _'SCSI 0 6 0 0 300 0 FLAMG-IO' to
                               _'rz3'

********************** ENTRY     2. ***************************

----- EVENT INFORMATION -----

EVENT CLASS                             ERROR EVENT
OS EVENT TYPE                199.       CAM SCSI
SEQUENCE NUMBER                1.
OPERATING SYSTEM                        DEC OSF/1
OCCURRED/LOGGED ON                      Tue Jan 11 18:05:10 1994
OCCURRED ON SYSTEM                      pearl
SYSTEM ID                x0004000F      CPU TYPE: DEC
                                        CPU SUBTYPE: KN15AA
----- UNIT INFORMATION -----

CLASS                         x0005     RODIRECT
SUBSYSTEM                     x0000     DISK
BUS #                         x0000
                              x0020     LUN x0
                                        TARGET x4
```

D.1 Specifying the Report Source

The following sections describe how to use the `uerf` command options that allow you to specify the source used to generate event reports.

D.1.1 Selecting the Event Class

Use the `uerf` command with the –`c` option to select the specified class of events. The `uerf` –`c` command has the following syntax:

uerf -c *class*

You can specify the following *`class`* variables:

Event Class	Description
`err`	Reports hardware-detected and software-detected events.
`maint`	Reports events that occur during system maintenance, such as running the online functional exercisers.
`oper`	Reports information on system status, autoconfiguration messages, device status and error messages, time stamps, and system startup and shutdown messages.

D.1.2 Selecting Disk Events

Use the `uerf` command with the –`D` option to select events for the specified disk type (for example, rz55) or disk class (for example, rz). The `uerf` –`D` command has the following syntax:

uerf -D [*disk*...]

If you do not specify a *`disk`* variable, events for all disks are reported. If you specify more than one *`disk`* variable, separate them with commas. For example:

/usr/sbin/uerf –D rz23,rz24

D.1.3 Selecting Mainframe Events

Use the `uerf` command with the –`M` option to select events for the specified mainframe event type. The `uerf` –`M` command has the following syntax:

uerf -M [*mainframe*...]

You can specify the following *`mainframe`* variables:

Mainframe Events	Description
`cpu`	Reports CPU-related events, such as machine checks.
`mem`	Reports memory-related events, such as single-bit corrected read data (CRD) and double-bit uncorrectable errors.

If you do not specify a *`mainframe`* variable, all mainframe events are reported. If you specify more than one *`mainframe`* variable, separate them with commas. For example:

/usr/sbin/uerf –M cpu,mem

D.1.4 Selecting Events As They Occur

Use the `uerf` command with the –`n` option to report events as they occur. You can use this option if you run the system exercisers. The `uerf` –`n` command has the following syntax:

uerf -n

You cannot specify the –`f` option with the –`n` option.

D.1.5 Selecting Operating System Events

Use the `uerf` command with the –O option to select operating system events such as panics, exceptions, and faults. The `uerf` –O command has the following syntax:

uerf -O [*op_system*...]

You can specify the following *op_system* variables:

Operating System Events	Description
aef	Arithmetic exception faults
ast	Asynchronous trap exception faults
pag	Page faults
pif	Privileged instruction faults
pro	Protection faults
ptf	Page table faults
raf	Reserved address faults
rof	Reserved operand faults
scf	System call exception faults
seg	Segmentation faults

If you do not specify an *op_system* variable, all operating system events are reported. If you specify more than one *op_system* variable, separate them with commas. For example:

```
# /usr/sbin/uerf –O raf,ptf,ast
```

D.1.6 Selecting Tape Events

Use the `uerf` command with the –T option to select events for the specified tape type (for example, tz30) or tape class (for example, tz). The `uerf` –T command has the following syntax:

uerf -T [[*tape*]]

If you do not specify a *tape* variable, events for all tape types and tape classes are reported. If you specify more than one *tape* variable, separate them with commas. For example:

```
# /usr/sbin/uerf -T tz
# /usr/sbin/uerf -T tz31
```

D.1.7 Generating Reports from Files

Use the `uerf` command with the –`f` option to select events from the specified log file instead of the default log file, which is defined by the `*.*` entry destination in the `/etc/binlog.conf` file. The `uerf` –`f` command has the following syntax:

uerf -f *filename*

The *filename* variable specifies the event-log file to use. You must specify the full pathname for the file, for example:

```
# /usr/sbin/uerf -f /var/adm/binary.old
```

You cannot specify the –`n` option with the –`f` option.

D.1.8 Generating Reports for Hosts

Use the `uerf` command with the –`H` option to select events for the specified host system. Use this option if events from remote systems are being forwarded to your local system. The `uerf` –`H` command has the following syntax:

uerf -H *hostname*

D.1.9 Selecting Events by Record Code

Use the `uerf` command with the –`r` option to select events with the specified record codes. The –`r` option offers an alternate way to report specific events, such as disk and tape events. The `uerf` –`r` command has the following syntax:

uerf -r *record...*

You can specify the following *record* variables:

Record code	Hardware-Detected Events
100	CPU machine checks and exceptions
101	Memory errors (soft and hard)
102	Disk errors
103	Tape errors
104	Device controller errors
105	Adapter errors
106	Bus errors

Record code	Hardware-Detected Events
107	Stray interrupts
108	Console events
109	Stack dump
199	CAM (SCSI) events
Record code	**Software-Detected Events**
201	ci ppd events
202	scs events
Record code	**ASCII Messages**
250	Informational
Record code	**Operational Messages**
300	Startup
301	Shutdowns and reboots
302	Panics
350	Diagnostics status

If you specify more than one *record* variable, separate them with commas. You can also separate *record* variables with a dash (–) to indicate a sequence of record codes.

The following example produces all system startup messages, including hardware devices configured and their control status register (CSR) addresses:

```
# /usr/sbin/uerf -r 300
```

The following example specifies a sequence of records:

```
# /usr/sbin/uerf –r 100-109
```

The following example specifies two records:

```
# /usr/sbin/uerf –r 100,102
```

D.2 Restricting Events

The following sections describe how to restrict the event selection in the report by specifying a time range, sequence numbers, or a unit number with the `uerf` command. You can also exclude events from a particular source.

D.2.1 Specifying Sequence Numbers

Use the `uerf` command with the –`s` option to select events with the specified sequence numbers. A sequence number is assigned to an event when it is logged. You can use this option to report specific events after viewing the event-log file at your terminal. The `uerf` –`s` command has the following syntax:

uerf -s *seq_of_numbers*

The *seq_of_numbers* variable specifies the beginning and ending sequence numbers separated by a dash (–). For example:

```
# /usr/sbin/uerf –s 750-800
```

Note

Sequence numbers restart when you reboot the system. If the event-log file contains events from before and after a reboot, the file may contain duplicate sequence numbers.

If the –`s` option is the only `uerf` command option specified, all events with the specified sequence numbers are reported.

D.2.2 Specifying a Time Range

Use the `uerf` command with the –`t` option to select events in the specified time range. The `uerf` –`t` command has the following syntax:

uerf -t *time*

The *time* variable specifies the start and end of the time range. If you do not use the –`t` *time* option, the entire event-log file is used to report events. The *time* variable has the following syntax:

s: *dd-mmm-yyyy*, *hh:mm:ss* **e:** *dd-mmm-yyyy*, *hh:mm:ss*

The *dd-mmm-yyyy* variable specifies the day, month, and year. The *hh:mm:ss* variable specifies the hours, minutes, and seconds. You specify the start time after the `s:` symbol, and you specify the end time after the `e:` symbol.

The `uerf` –`s` command uses the following defaults for the date and time:

- The current date
- The start time is 00:00:00
- The end time is 23:59:59

The following example produces a report that contains all events for the 24-hour period of January 11, 1994:

```
# uerf –t s:11-jan-1994,00:00:00 e:11-jan-1994,23:59:59
```

The following command produces a report from the beginning of the event-log file until February 29 of the current year:

```
# /usr/sbin/uerf –t e:29-feb
```

The following command produces a report for all events on the current day and year, starting at 1:20 p.m. and ending at the current time:

```
# /usr/sbin/uerf –t s:13:20
```

D.2.3 Specifying Unit Numbers

Use the `uerf` command with the –u option to select events from the disk or tape device unit number. The `uerf` –u command has the following syntax:

uerf -u *number*

The *number* variable specifies the tape or disk unit number. You can use this option only with the –D and the –T options.

D.2.4 Excluding Reported Events

Use the `uerf` command with the –x option to exclude the specified event source from the report. You can exclude the –c, –D, –M, –O, and –T options from the request. Refer to Section D.1 for more information on event sources. The `uerf` –x command has the following syntax:

uerf -x [-c] [-D] [-M] [-O] [-T]

The options to be excluded can appear before or after the –x option.

For example, the following command reports all events except disk events and operating system events:

```
# /usr/sbin/uerf –O –x –o full –D
```

D.3 Controlling the Report Output

The following sections describe the options that control the report output of the `uerf` command.

D.3.1 Generating Summary Reports

Use the `uerf` command with the –S option to produce a summary report. All the `uerf` source selection options (–c, –D, –M, –O, and –T) support

summaries. The default format for summary report output is terse. Refer to Section D.3.2 for more information on output formats.

The following example shows the command and options that generate a terse summary of all events recorded for the day in the log file:

```
# /usr/sbin/uerf -t s:00 -S
```

D.3.2 Specifying the Type of Output

Use the `uerf` command with the –`o` option to format the report output. The `uerf` –`o` command has the following syntax:

uerf -o *output*

The *output* variable can be one of the following:

Output Type	Description
`brief`	Reports event information in a short format. This is the default.
`full`	Reports all available information for each event.
`terse`	Reports event information and displays register values but does not translate the events to ASCII.

Usually, the –`o full` option produces the most event information. However, panic messages and other ASCII messages do not provide more information with the –`o full` option.

The following example shows the default brief format for a memory event:

```
# /usr/sbin/uerf -r 101
```

The following example shows the information produced by full output format for this report, which displays all memory-related events:

```
# /usr/sbin/uerf –o full -r 101
```

D.3.3 Generating Reports in Reverse Chronological Order

Use the `uerf` command with the –`R` option to report events in reverse chronological order.

The following example causes the `uerf` command to produce a report that lists all startup messages, beginning with the most recent:

```
# /usr/sbin/uerf –R –r 300
```

D.3.4 Displaying Hexadecimal Output

Use the `uerf` command with the –Z option to output event entries in hexadecimal format.

E

Administering Specific Hardware Devices

E.1 Introduction

This appendix describes the procedures for adding and configuring certain hardware devices. Current supported devices are:

- PCMCIA cards
- CalComp graphics tablet

E.2 PCMCIA Support

PCMCIA (PC Card) support is limited to the following capabilities:

- Support of selected ISA to PCMCIA bridge adapters
- Support on the following platforms:
  - AlphaStation 255
  - AlphaStation 200
  - AlphaStation 400
  - AlphaStation 600
  - AlphaServer 1000
- One modem card, specifically Megahertz XJ2288 (28.8kpbs)
- Hot swap capability of PC Cards

E.2.1 Restrictions

The following restrictions apply in this release.

- No support is provided for loadable device drivers for PC Cards.
- If the system does not have enough available IRQ (interrupt) numbers to assign to the PCMCIA devices, PCMCIA devices cannot be configured. To support one PCMCIA adapter the system must have at least three unused IRQ numbers available. One IRQ is for the adapter and the other two are for each PCMCIA socket.

- Digital UNIX can support two PCMCIA adapters in a system provided that the necessary resources are available. In some systems, availability of interrupt lines will prohibit the use of multiple adapters. If you have sufficient resources and are going to support two adapters, the second adapter should be configured to use the I/O address 3E2.
- To use fax functions in a fax/modem PC card, a commercial UNIX fax application software program is required.
- The Megahertz XJ2288 is the only modem card fully qualified on Digital UNIX. However, other modem cards of similar type (both 14.4kpbs and 28.8kpbs) may work. The following is the list of modem cards that are known to work:
  - Model XJ2288, from MEGAHERTZ
  - Model XJ1144, from MEGAHERTZ
  - KeepInTouch Cardcard from AT&T Paradyne
  - PCMCIA V.32bis 14,400 Fax from Digital Equipment Corporation
- The selected ISA to PCMCIA bridge adapters are from SCM Microsystems. The SWAPBOX CLASSIC X2 Model MMCD-D2 which has the following features:
  - 3.5 inch front access
  - Two slots (type II + type III) PC card socket
  - Standard PC-AT 16-bit ISA bus interface
  - PCMCIA Revision 2.X and ExCA compliant

  The SWAPBOX PREMIUM COMBO Model MMCD-FC2 has the following features:
  - 3.5 inch, 1.44 Mbyte Floppy Drive Support.
  - One Type I, II, or III front-access PC card socket
  - One Type I, II, or III rear-access PC card socket
  - Standard PC-AT 16-bit ISA bus interface
  - PCMCIA Revision 2.X and ExCA compliant

  However, other ISA to PCMCIA bridge adapters using the Intel i82365SL or a compatible device may also work.

E.2.2 Configuring the PCMCIA Adapter Board from the Console

Before inserting the PCMCIA adapter board into your system, make sure to read the manual that came with the adapter from the adapter vendor and follow the instructions on how to connect the cables and install the board. Check your system documentation to find out what kind of bus is

available in your system and use the appropriate ISA or EISA instructions in this section.

E.2.2.1 Configuring on an ISA Bus System

1. If the system is an ISA bus system, the `isacfg` utility from the console must be used to configure the PCMCIA adapter.
2. After the PCMCIA adapter board is inserted to an ISA slot in the system, turn on the system.
3. To add an PCMCIA option to the platforms with an ISA bus, issue the following ISA option card configuration command at the console. The following example uses an AlphaStation 200 platform, but the commands should be the same in all three ISA bus platforms.

   ```
   >>> isacfg -slot 1 -etyp 1 -dev 0 -mk -iobase0 3e0 \
   -irq0 14 -enadev 1 -handle PCIC-PCMCIA
   ```

   If the system is already using slot 1, select and unused slot number.
4. The IRQ (interrupt) number must not conflict with interrupt numbers that are assigned to other default devices on the system. The system hardware manual usually indicates which IRQ numbers are assigned to default devices.
5. The recommended IRQ number for the PCMCIA adapter is 14 (decimal).
6. If IRQ 14 is already used, the next best choice is IRQ 10, if it is not already used by other devices.
7. When you issue the above `isacfg`command, the console should print out the following line or something similar:

   ```
   type >>>init to use these changes
   ```
8. After reinitializing the console, you can verify that you configured the PCMCIA adapter correctly by issuing the following command:

   ```
   >>>isacfg -slot 1
   ```
9. You should see the following screen display:

   ```
   ============================================================
   handle: PCIC-PCMCIA
   etyp: 1
   slot: 1  dev: 0
   enadev: 1
   totdev: 1
   iobase0: 3e0  membase0: 8000000000000000
   iobase1: 8000000000000000  memlen0: 8000000000000000
   iobase2: 8000000000000000  membase1: 8000000000000000
   iobase3: 8000000000000000  memlen1: 8000000000000000
   iobase4: 8000000000000000  membase2: 8000000000000000
   iobase5: 8000000000000000  memlen2: 8000000000000000
   ```

```
rombase: 8000000000000000
romlen: 8000000000000000
dmamode0/chan0: 80000000 irq0: 14
dmamode1/chan1: 80000000 irq1: 80000000
dmamode2/chan2: 80000000 irq2: 80000000
dmamode3/chan3: 80000000 irq3: 80000000

============================================================
>>>
```

E.2.2.2 Configuring on an EISA Bus System

If you are installing the PCMCIA adapter on an EISA bus system, use the EISA Configuration Utility (ECU) to configure it. Invoke the EISA Configuration Utility (ECU), and specify that the PCMCIA adapter is present. Next, provide a pointer to the `aisa3000.cfg` configuration file.

Refer to your system hardware documentation for complete instructions on how to run the ECU program.

E.2.3 Configuring and Using a PCMCIA Modem PC Card

Since a PC Card is a dynamic device (i.e. not a static device that is present all the time in the system hardware), and the serial-line device driver is a static device driver, when the system is installed initially, there will not be a corresponding `acex` entry created automatically by the `doconfig` of the target system. This is due to the fact that the system does not know when it is being installed that there will be a fax/modem card for PCMCIA since the card is not in the system yet.

If you want the system to automatically create the `acex` entry for your PCMCIA fax/modem card, before you start installing the system, make sure that you have the PCMCIA adapter configured in the console and that the PCMCIA fax/modem card is inserted into the slot. If you have a fax/modem card in the slot 0, for example, when the system is installed and the target kernel is built, the system kernel configuration file built will have the following entry:

```
controller ace2   at pcmcia0  slot 0 vector aceintr
```

The installation will also create the device special file for this fax/modem card in the directory named `/dev`.

```
# ls -gl tty02

crw-rw-rw-   1 root     system    35,  2 Oct 16 13:22 tty02
```

If you did not have the PCMCIA fax/modem card inserted in the slot when the system was installed, then you need to add the following line to your system kernel configuration file, (`/sys/conf/HOSTNAME` where `HOSTNAME` is the name of your system):

```
controller              ace2    at *    slot ? vector aceintr
```

If you plan to use two modem cards simultaneously, add the following lines to your system configuration file:

```
controller              ace2    at *    slot ? vector aceintr
controller              ace3    at *    slot ? vector aceintr
```

Once the system configuration file is modified, use the following command to rebuild the new kernel and reboot the system.

```
# doconfig -c
```

E.2.4 Creating a Device Special File for the Modem Card

Normally the system installation creates the following two default `tty0x` device special files in the directory `/dev`:

```
crw-rw-rw-   1 root     system    35,  0 Oct 16 13:22 tty00
crw-rw-rw-   1 root     system    35,  1 Oct 16 13:22 tty01
```

This is because most systems have two embedded serial lines. A system with a single embedded serial line creates only one`tty00` entry in the `/dev` directory.

To create additional device special files for the PCMCIA modem cards, use the MAKEDEV utility in the `/dev` directory. For example:

```
# ./MAKEDEV ace2

MAKEDEV: special file(s) for ace2:
tty02
```

The generated special file should look like this:

```
crw-rw-rw-   1 root     system    35,  2 Oct 27 14:02 tty02
```

If you intend to have two PCMCIA modem cards working simultaneously, create device special files for each card. For example:

```
# ./MAKEDEV ace2 ace3

MAKEDEV: special file(s) for ace2:
tty02
MAKEDEV: special file(s) for ace3:
tty03
```

The generated special files should look like this:

```
crw-rw-rw-   1 root     system    35,  2 Oct 27 14:02 tty02
crw-rw-rw-   1 root     system    35,  3 Oct 27 14:02 tty03
```

E.2.5 /etc/remote File

You must edit the `/etc/remote` file must be modified to add new access line definitions for the PCMCIA modem cards to be used. If you have a 28.8kpb modem card and will be using the full speed, the baud rate (br) in the `/etc/remote` file should be set to 38400.

For example, add the following line to the `/etc/remote` file:

```
line2:dv=/dev/tty02:br#38400:pa=none:
```

Note that `line2` can be any name you determine to be used with the `tip` command to establish a connection.

Once the PCMCIA modem card is inserted correctly and the system configures the card, the card can be used the same as any other modem devices.

E.2.6 Inserting a PCMCIA Modem Card

To use a PCMCIA modem card, insert the card to one of the PC Card slots in the PCMCIA adapter. Depending on the adapter type, there may be two front access card slots or one front access and one rear access card slot. When you insert the card into the slot 0, you should see the following message on the console terminal (or the Console Log window of the graphics head).

```
# PCMCIA socket 0: card manufacturer: MEGAHERTZ
product name: XJ2288
Configured: serial unit 2, type=16550A
ace2 at pcmcia0
```

This example used the MEGAHERTZ XJ2288 fax/modem card.

When a modem card is inserted, an error message such as the following may appear on the Console Log window:

```
socket 0: card manufacturer: MEGAHERTZ, unknown modem card inserted

Using generic modem driver for this PC Card.

PCMCIA socket 0: card manufacturer: MEGAHERTZ

product name: XJ1144

socket 0: Couldn't find usable config. for this card.
Please eject this PC Card.
```

This error occurs if the card requires I/O resources that are already in use by other components in the system. If this error message is seen, the card should be ejected, because it is not configured. A possible solution is to remove some other ISA/EISA devices in the system and reboot the system, freeing I/O resources that may be required.

E.2.7 Removing a PCMCIA Modem Card

Once you are finished using the modem card, push the button next to the card slot to eject it. You should see the following message on the console terminal or console Log window.

```
# stray interrupt on unit=2, intr_id=0
PCMCIA socket 0: PC Card removed
```

This message is not always displayed when you eject the card. It only happens if the serial line driver generates an interrupt when the card got ejected.

E.3 CalComp Graphics Tablet

This section provides information about how configure a CalComp DrawingBoard III tablet, an input device supported by the Xinput extension to the Xserver. Once the software for the tablet is installed on your system, you can configure it to emulate a system mouse.

E.3.1 Configuring the CalComp DrawingBoard III Tablet

If you intend to use the CalComp DrawingBoard III tablet software, you must edit the file `/usr/var/X11/Xserver.conf` to turn on support for the X Input extension. To do this, remove the comment characters surrounding the following lines:

```
input <
<_dec_xi_db3  lib_dec_xi_db3.so  XiDb3Init /dev/tty00:1:12:12:16:\
1:8:1000:1:1 >
>
```

The backslash in this example indicates line continuation and is not in the actual display.

You should also review these lines to ensure that the options specified for the tablet are correct, especially that the `tty` that is specified as the serial port where the tablet is connected to your system.

The last line of this file has the following syntax:

```
device:mode:tabletWidth:tabletHeight:numbtns:corePointer:mouseScale:\
resolution:Xincrement:Yincrement
```

The backslash in this example indicates line continuation and is not in the actual display.

Table E–1 can help you determine how to set up the entries for the tablet in the `/usr/var/X11/Xserver.conf` file.

Table E–1: CalComp DrawingBoard III Tablet Configuration Options and Values

Option	Description
device	The port (`tty`) to which the device is connected. The default is tty00.
mode	This should be set to 1 for absolute motion.
tabletWidth	Width of the active tablet area in inches, not the physical size. The default is 12.
tabletHeight	Height of the active tablet area in inches, not the physical size. The default is 12.
numbtns	Number of buttons on the puck or pen. The maximum number is 16 and the default is 16.
corePointer	0 indicates a native tablet mode (no system mouse). 1 indicates emulate core pointer (the mouse and tablet are both core pointer devices. The default is 1 (emulate core pointer).
mouseScale	1 to 50 scaling factor in relative mode. Determines the speed of the cursor; the higher the number, the slower the cursor moves. The default is 8.
resolution	1 to 2540 lines per inch (lpi). The default is 1000.
Xincrement	How much the X axis must be incremented to cause the tablet to send new coordinates to the Xserver. The range is 0 to 65536. The default is 1.
Yincrement	How much the Y axis must be incremented to cause the tablet to send new coordinates to the Xserver. The range is 0 to 65536. The default is 1.

The device option is required and specifies which `tty` device should be associated with the tablet. By default, the installation software assigns the CalComp DrawingBoard III tablet to tty00, which you may want to change if that `tty` is already allocated. For information on how to determine which serial port your tablet is connected to, see the hardware documentation that was shipped with your processor.

Note that when the stylus or puck is moved as far as the minimum Xincrement or Yincrement value, the value of the corresponding axis is updated. For example, if the Xincrement value is set to 10 and the tablet is moved 10 units along the X axis, the value of the Y axis will also be updated simultaneously with the X axis, even if the Yincrement value has not been reached. Keep this in mind when setting the Xincrement and Yincrement options.

After you have configured the /usr/var/X11/Xserver.conf file, you must follow these steps to turn on support for the tablet in the Xserver:

1. Plug the tablet into your system and turn it on.
2. Enter the following command to restart the Xserver so that the Xinput extension can recognize the tablet:

   ```
   # /usr/sbin/shutdown -r +5 \
   "Turning on support for the Calcomp Drawingboard III tablet"
   ```

   (The backslash in this example indicates line continuation and is not in the actual display.) When the system comes back up, the tablet will be configured into the Xserver and ready to use.

When the Xserver first accesses the tablet, it performs some hardware-specific initialization that can be saved in the on-board memory of the tablet. To save these settings, follow these steps:

1. Press the EXIT CONFIG button on the tablet's menu.
2. Under the SAVE button, press the DEFAULT button.
3. Press the EXIT CONFIG button to save the settings.

E.3.2 Notes and Restrictions

The following notes and restrictions apply to the CalComp DrawingBoard III tablet:

- If the puck or stylus is not used within a 5 minute period, the tablet will automatically shut off. To reactivate it, press any button on the puck or stylus while they are in close proximity to the tablet.
- If you configure the tablet as the system's core pointer, moving the puck and the system mouse simultaneously will cause the cursor to move in an unpredictable fashion.
- Use only one puck or stylus at a time. If you try to use both input devices simultaneously, you will encounter unpredictable behavior.
- Digital UNIX does not support manual configuration of the tablet via the tablet buttons. If you try to use these buttons to configure the tablet, the Xserver will malfunction and may even crash. If you need to reconfigure the tablet, edit the /usr/var/X11/Xserver.conf file and then reboot the Xserver.
- You can modify some parts of the tablet setup by programming the Xinput extension. For more information on how to do this, see the XInput specification provided by the X Consortium.

Index

A

B

C

D

E

F

G

H

I

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Y

Z

How to Order Additional Documentation

Technical Support

If you need help deciding which documentation best meets your needs, call 800-DIGITAL (800-344-4825) before placing your electronic, telephone, or direct mail order.

Electronic Orders

To place an order at the Electronic Store, dial 800-234-1998 using a 1200- or 2400-bps modem from anywhere in the USA, Canada, or Puerto Rico. If you need assistance using the Electronic Store, call 800-DIGITAL (800-344-4825).

Telephone and Direct Mail Orders

Your Location	Call	Contact
Continental USA, Alaska, or Hawaii	800-DIGITAL	Digital Equipment Corporation P.O. Box CS2008 Nashua, New Hampshire 03061
Puerto Rico	809-754-7575	Local Digital subsidiary
Canada	800-267-6215	Digital Equipment of Canada Attn: DECdirect Operations KAO2/2 P.O. Box 13000 100 Herzberg Road Kanata, Ontario, Canada K2K 2A6
International	—	Local Digital subsidiary or approved distributor
Internal (submit an Internal Software Order Form, EN-01740-07)	—	SSB Order Processing – NQO/V19 *or* U.S. Software Supply Business Digital Equipment Corporation 10 Cotton Road Nashua, NH 03063-1260

Reader's Comments

Digital UNIX
System Administration
AA-PS2RE-TE

Digital welcomes your comments and suggestions on this manual. Your input will help us to write documentation that meets your needs. Please send your suggestions using one of the following methods:

- This postage-paid form
- Internet electronic mail: `readers_comment@zk3.dec.com`
- Fax: (603) 881-0120, Attn: UEG Publications, ZK03-3/Y32

If you are not using this form, please be sure you include the name of the document, the page number, and the product name and version.

Please rate this manual:

	Excellent	Good	Fair	Poor
Accuracy (software works as manual says)	☐	☐	☐	☐
Clarity (easy to understand)	☐	☐	☐	☐
Organization (structure of subject matter)	☐	☐	☐	☐
Figures (useful)	☐	☐	☐	☐
Examples (useful)	☐	☐	☐	☐
Index (ability to find topic)	☐	☐	☐	☐
Usability (ability to access information quickly)	☐	☐	☐	☐

Please list errors you have found in this manual:

Page	Description
_____	_____
_____	_____
_____	_____
_____	_____

Additional comments or suggestions to improve this manual:

_____
_____
_____
_____

What version of the software described by this manual are you using? _____

Name, title, department _____
Mailing address _____
Electronic mail _____
Telephone _____
Date _____

Do Not Cut or Tear − Fold Here and Tape

digital™

NO POSTAGE
NECESSARY IF
MAILED IN THE
UNITED STATES

BUSINESS REPLY MAIL
FIRST−CLASS MAIL PERMIT NO. 33 MAYNARD MA

POSTAGE WILL BE PAID BY ADDRESSEE

DIGITAL EQUIPMENT CORPORATION
UEG PUBLICATIONS MANAGER
ZKO3−3/Y32
110 SPIT BROOK RD
NASHUA NH 03062−9987

Do Not Cut or Tear − Fold Here

Cut on
Dotted
Line