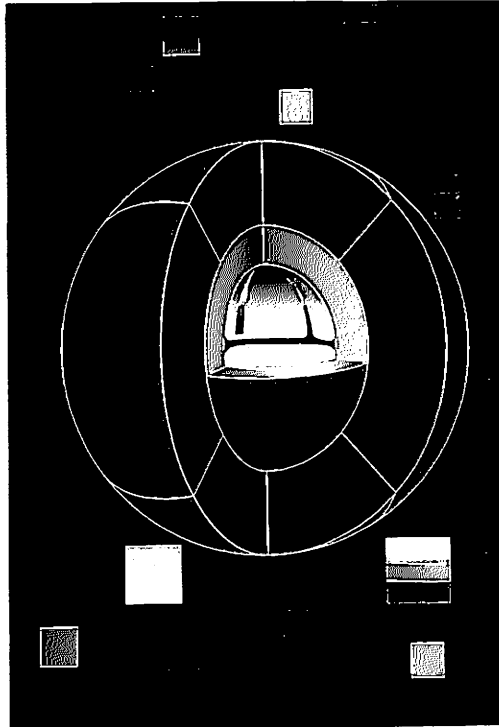


NCR System 3000

NCR UNIX SVR4 MP-RAS



Administrator Guide:
Command Line Interface
General Administration (Volume 1)
Release 2.02
September 1993
D1-2266-D

The program product(s) described in this book is a licensed product of NCR Corporation.

UNIX is a registered trademark of UNIX Systems Laboratories.

It is the policy of NCR Corporation to improve products as new technology, components, software, and firmware become available. NCR Corporation, therefore, reserves the right to change specifications without prior notice.

All features, functions, and operations described herein may not be marketed by NCR in all parts of the world. In some instances, photographs are of equipment prototypes. Therefore, before using this document, consult your NCR representative or NCR office for information that is applicable and current.

Copyright © 1993
By NCR Corporation
Dayton, Ohio U.S.A.
All Rights Reserved
Printed in U.S.A.
Confidential, unpublished
Property of NCR Corporation

To maintain the quality of our publications, we need your comments on the accuracy, clarity, organization, and value of this book. Address correspondence to:

Information Products
NCR Corporation
3325 Platt Springs Road
West Columbia, SC 29170
U.S.A.

Preface

Who Should Read This Book

This book is for the knowledgeable system administrator, that is, the person who administers the system using the operating system commands at the shell prompt (# for the **root** login).

Releases Covered In This Book

This book applies to Release 2.02 of the operating system for your NCR System 3000 computer.

What You Should Know

You should be familiar with the information in the following books:

- *User Guide: Introduction to NCR UNIX SVR4*
- *NCR UNIX SVR4 MP-RAS Administrator Guide: OA&M Menu Interface*

How To Use This Book

This book assumes you are familiar with the *NCR UNIX SVR4 MP-RAS Administrator Guide: OA&M Menu Interface* and takes you one step beyond that book to include:

- Command line procedures that perform some of the same tasks that the menu procedures perform in the *NCR UNIX SVR4 MP-RAS Administrator Guide: OA&M Menu Interface*

- Command line procedures for many more advanced tasks
- Diagnostics and trouble-shooting procedures

Rather than lead you through chapter by chapter, this book provides a reference in which you can look up a particular topic or task. Use the *NCR UNIX SVR4 MP-RAS Reference Manual* as needed to obtain additional information about a particular command.

This book consists of the following volumes:

Volume 1 — General Administration

This volume describes the following:

- System administration logins, duties, and schedules
- Changing run levels
- Using the Reference, Diagnostics, and boot flex diskettes
- Recovery procedures using the Maintenance (*/install*) file system
- Administering, checking, and repairing the file system
- Backing up and restoring information

Volume 2 — Devices and Networks

This volume describes the following:

- Installing additional disks, adapters, and terminals
- Setting up and managing the LP Print Service
- Setting up and administering the Basic Networking Utilities (BNU)

Volume 3 — System Configuration

This volume describes the following:

- The files and directories used in configuration
- The process of reconfiguring the system and building new kernels
- The tunable configuration parameters

Appendices and Glossary

The appendices describe the file system, device names and numbers, customizing the sysadm interface, using national characters on the console, and using sendmail. The glossary defines terms you encounter in this book.

Conventions Used In This Book

- The following type identifies text that you must enter exactly as shown:

```
login su
```

- The following type identifies output from a screen or command:

```
Press the Return key to continue.
```

- Path names and file names appear in italics. For example:

The */etc/profile* file defines the standard environment for all users.

- Configuration parameters and shell variables appear in capital letters. Capital letters are used for emphasis also. For example:

Press RETURN after specifying a value for the TERM variable. Do NOT leave this value blank.

- Utilities, commands, user names, and terminal keys appear in boldface type. For example:

The **cpio**(1) command backs up files.

- Parameters for command line options appear in italics within the text. For example:

Specify the class (*-cclass*) for the printer.

- The numbers and letters in parentheses after a command name indicate the type of command.

(1)

Standard command available to users at the \$ prompt.

(1M)

Command available only to someone logged in as **root**

(2)

System call

(3)

Library function

Related Publications

For supplementary information, refer to the following books:

- *User Guide: Introduction to NCR UNIX SVR4*
- *NCR UNIX SVR4 MP-RAS Administrator Guide: OA&M Menu Interface*
- *NCR UNIX SVR4 Message Manual*
- *NCR 345x/35xx Unit Installation, Care, and Cleaning*
- *NCR UNIX SVR4 MP-RAS Reference Manual*

Contents

Volume 1 — General Administration

Chapter 1	Common System Administration Tasks	
	Accessing the System	1-2
	Changing Run Levels	1-5
	Starting Up and Shutting Down the System	1-7
	Specifying Run Levels During System Startup	1-9
	Setting Up Your System for Automatic or Manual File System Checks	1-12
	Setting up an Alternate Boot Disk	1-17
	Maintaining the Password and Shadow Files	1-28
	Managing Software Packages	1-30
	Managing Multiple Processors	1-32
	Setting the Power Fail Strategy	1-37
	Customizing Memory Dumps	1-44
	Installing from a Remote Console (NCR 3450/3550 Only)	1-47

Chapter 2

Using the Maintenance and Reference Diskettes

What Are Maintenance and Reference Diskettes?	2-2
Booting from a Flex Diskette	2-4
Copying Maintenance and Reference Diskettes	2-6
Accessing the Maintenance File System	2-11

Chapter 3

Recovery Techniques

Recovering Files in the /stand File System	3-2
Recovering unix	3-6
Recovering the root Password	3-10
Recovering the /etc/passwd File	3-13
Recovering the /etc/shadow File	3-17
Recovering Files from the Installation Tape	3-21
Recovering Sector 0	3-25
Recovering the Boot Loader Program	3-29
Saving Memory to Dump Area	3-35
Saving Dump Area to Media	3-37
Examine the Contents of the Dump	3-39
Checking a File System from the Maintenance File System	3-40

Chapter 4

Administering the File System

General Administration of File Systems	4-2
Making (Creating) File Systems	4-7
Mounting File Systems	4-11
Unmounting File Systems	4-13

Displaying File System Information	4-15
Compacting Files in /stand	4-18
Using Quotas	4-20

Chapter 5

Checking and Repairing the File System

Minimizing File System Corruption	5-2
Using fsck to Check and Repair the File System	5-4
Checking and Repairing s5 File Systems	5-8
s5 File System Components Checked by fsck	5-11
Phases of fsck on s5 File Systems	5-19
Checking and Repairing ufs File Systems	5-38
ufs File System Components Checked by fsck	5-40
Phases of fsck on ufs File Systems	5-47
Checking bfs File Systems	5-78

Chapter 6

Backing Up Information

What You Should Know Before You Perform a Backup	6-2
Backing Up File Systems	6-4
Backing Up Individual Files and Directories	6-10
Backing Up Raw Devices	6-14
Backing Up File System and Disk Information	6-16
Backing Up the Entire System	6-19
Backing Up an Entire NCR 3600	6-20
Copying Files to NCR TOWERS	6-25
Retensioning a Cartridge Tape	6-26

Chapter 7**Restoring Information**

Restoring File Systems	7-2
Restoring Individual Files and Directories	7-5
Restoring Raw Devices	7-8
Restoring an Entire Disk	7-10
Restoring the Entire System	7-20
Restoring a Corrupted Root File System	7-21
Restoring an Entire NCR 3600	7-27

Chapter 8**Administering the UNIX SVR4 Mail Subsystem**

Understanding SVR4 Mail	8-2
Establishing a Smarter Host	8-5
Establishing Domain Addresses	8-6
Establishing a Mail Cluster or Gateway	8-7
Establishing Mail Service on a Networked File System (RFS or NFS)	8-8
Administering Alias Lists	8-11
Other Surrogate File Routines	8-12
Enabling SMTP	8-14
Stopping and Starting the SMTP Daemon	8-17
Disabling SVR4 Mail	8-18
Setting Up SMTP to Listen Over Multiple Networks ..	8-20
Reading the SMTP Log File	8-21

Appendix A

The File System

The File System Hierarchy	A-2
Symbolic Links	A-5
File System Organization	A-6
File System Types	A-8
The s5 File System Type	A-10
The ufs File System Type	A-17
The bfs File System Type	A-23
The cdfs File System Type	A-27

Appendix B

Device Names and Numbers

Major and Minor Device Numbers	B-2
Device Names and Minor Numbers	B-5

Appendix C

Customizing the sysadm Interface

Interface Structure: a Hierarchy of Menus	C-2
Writing Your Help Messages	C-8
Creating or Changing a Menu Entry	C-18
Creating or Changing a Task Entry	C-25
Deleting a Menu or Task Entry	C-31

Appendix D**Using National Characters on the Console**

Changing Line Characteristics to 8 Bit	D-2
Changing Keyboard Mapping	D-4
Changing Screen Mapping	D-6

Appendix E**Sendmail – An Internetwork Mail Router**

Abstract	E-2
Design Goals	E-5
Overview	E-8
Usage and Implementation	E-13
Comparison with other Mail Programs	E-19
Evaluations and Future Plans	E-22
References	E-25

Appendix F**Sendmail Installation and Operation**

Introduction	F-2
Basic Installation	F-4
Normal Operations	F-10
Arguments	F-19
Tuning	F-21
The Configuration File	F-27
Command Line Flags	F-48
Configuration Options	F-50
Mailer Flags	F-55
Other Configuration	F-58

Summary of Support Files	F-66
Error/Status Messages	F-68

Volume 2 — Devices and Networks

Chapter 1

Configuring Disks

Using Disk Utilities	1-2
Adding Disks	1-3
Formatting Disks	1-6
Partitioning Disks	1-8
Writing Boot Code and Initializing Disks	1-13
Slicing Disks	1-14
Marking Bad Blocks	1-19
Displaying Slicing Information	1-23
Changing the Active Partition	1-24
Configuring Swap Space	1-25
Configuring the Dump Area	1-31
Replacing a Failed Disk	1-35

Chapter 2

Configuring Adapters

Configuring Adapters at System Startup	2-2
Configuring Adapters During System Operation	2-11
Installing a Serial Controller	2-18
Installing an Ethernet LAN Adapter	2-23
Adding a SCSI Controller to a Uniprocessor System ...	2-27
Adding a SCSI Controller to a Multiprocessor System .	2-34

Chapter 3

Configuring Serial Devices

The Service Access Facility	3-2
The Service Access Controller	3-3
The Port Monitor ttymon	3-8
Configuring a Serial Port for a Terminal	3-13
Configuring a Serial Port for a Modem	3-15
Configuring the EIA Mode for a Serial Port	3-18
Configuring a Second Serial Port	3-20
Configuring Serial Controller Ports for Terminals	3-22
Configuring a Serial Controller for Modems	3-25
Setting Modem Options	3-29
Enabling Dial-up Password Protection	3-32
Configuring a Serial Port for a Three-Wire Cable	3-35

Chapter 4

Using The LP Print Service

What is the LP Print Service?	4-2
Getting Started	4-10
Installing the LP Print Service	4-11
Adding Printers	4-12
Enabling Remote Printing	4-17
Managing Printer Classes	4-25
Setting Other Printing Options	4-27
Displaying the Status of Printers	4-37
Starting and Stopping the LP Print Service	4-39

	Managing the Printing Load	4-41
	Troubleshooting	4-49
Chapter 5	Managing the LP Print Service	
	Managing Queue Priorities	5-2
	Managing Printer Faults	5-7
	Providing Forms	5-12
	Providing Character Sets or Print Wheels	5-24
	Providing Filters	5-32
	Cleaning Out the Request Log	5-46
Chapter 6	Adjusting the LP Print Service	
	PostScript Printers	6-2
	Customizing the Print Service	6-12
Chapter 7	Network Selection	
	Network Selection	7-2
	Name-to-Address Mapping	7-10
Chapter 8	Basic Networking Utilities	
	General Description of the Basic Networking Utilities	8-2
	Commands	8-5
	Networking Daemons	8-8
	Supporting Data Base	8-10
	Administrative Support Files	8-12
	Logs	8-16

Chapter 9	Managing BNU	
	Basic Procedures	9-2
	Procedure for Setting Up a Link	9-5
	Establishing the Physical Connection	9-7
	Setting the Node Name of the Running Kernel	9-9
	Identifying Systems for Communication	9-11
	Adding uucp logins	9-20
	Identifying Devices Used for Communication	9-21
	Writing Dialing Information	9-30
	Creating Access and Security Mechanisms	9-35
	Editing Additional Files	9-48
	Specifying File Transfer Protocols	9-60
	Verifying the Link To Be Sure It Works	9-63
	Maintaining BNU	9-64
	Common Problems	9-71
	Troubleshooting with cu	9-73
	Troubleshooting with Uutry	9-74
	Using Other Troubleshooting Tools	9-77
	Basic Networking Utilities Error Messages	9-78

Volume 3 — System Configuration

Chapter 1

Monitoring System Activity

Using Monitoring Utilities	1-2
Collecting Activity Data with <code>sadc</code>	1-5
Analyzing Activity Data with <code>sar</code>	1-7
Analyzing Activity with Process Accounting	1-35
Analyzing Specific Processes with <code>time</code>	1-45
Analyzing Specific Processes with <code>timex</code>	1-47
System Profiling	1-49

Chapter 2

Process Scheduling

Using the Process Scheduler	2-2
Configuring the Scheduler	2-8
Changing Scheduler Parameters	2-21

Chapter 3

Configuring Kernels

Configuration Files and Directories	3-2
Creating New Kernels	3-11
Loading/Booting the Kernel	3-15
Modules	3-17
Configuration Parameters	3-25

Chapter 4

Configuration Parameters

Chapter 5	Tuning Basics	
	Reasons for Tuning	5-2
	Basic Steps	5-3
	Knowing When It's Time To Tune	5-6
	System Resources	5-7
	Potential Bottlenecks	5-10
	Basic Tuning Procedure	5-13
Chapter 6	Memory Utilization	
	Memory Management	6-2
	Demand Paging	6-3
	Swapping	6-7
	Flushing	6-10
Chapter 7	Disk Utilization	
	Factors Affecting Disk Performance	7-2
	I/O Balancing	7-3
	Buffer Utilization	7-4
	Monitoring and Controlling Disk Usage	7-7
	Directory Reorganization	7-14
	Eliminating File System Fragmentation	7-16

Chapter 8	CPU Utilization	
	Workload and CPU Time	8-2
	Efficient PATH Variables	8-3
	Excessive Pathname Searches	8-5

Figures

Volume 1 — General Administration

Figure 1-1:	Run Levels (1 of 2)	1-5
Figure 1-2:	Run Levels (2 of 2)	1-6
Figure 1-3:	Relationships Between /etc/passwd and /etc/shadow	1-29
Figure 1-4:	Power Backup System Support Hardware	1-39
Figure 4-1:	Possible File System Block Sizes	4-8
Figure 5-1:	ufs File System Specific Options	5-39
Figure 8-1:	Status Values	8-24
Figure 8-2:	Error Codes	8-27
Figure A-1:	The Root File System	A-3
Figure A-2:	Root File System Contents	A-4
Figure A-3:	A UNIX File System	A-6
Figure A-4:	Adding the /usr File System	A-7

Figures

Figure A-5:	The UNIX View of an s5 File System .	A-11
Figure A-6:	The File System Address Chain for s5	A-15
Figure A-7:	The UNIX View of a ufs File System .	A-18
Figure A-8:	The File System Address Chain in a ufs File System	A-21
Figure A-9:	The UNIX View of a bfs File System .	A-23
Figure B-1:	Major and Minor Numbers in BASE O.S. (1 of 3)	B-2
Figure B-2:	Major and Minor Numbers in BASE O.S. (2 of 3)	B-3
Figure B-3:	Major and Minor Numbers in BASE O.S. (3 of 3)	B-4
Figure B-4:	Minor Number Bit Definitions	B-8
Figure B-5:	SCSI Device Name Components	B-9
Figure B-6:	Example 1 of SCSI Device Name and Number	B-10
Figure B-7:	Example 2 of SCSI Device Name and Number	B-10
Figure C-1:	Sysadm Main Menu	C-2
Figure C-2:	Sysadm Machine Menu	C-3
Figure C-3:	Item Help File for One Form	C-15
Figure C-4:	FACE Form Associated with Item Help File in Figure C-3	C-15

Figure C-5:	Item Help File for Multiple Forms	C-16
Figure C-6:	Item Help File for Multiple Forms	C-17
Figure C-7:	Sample Menu Definition Form	C-24
Figure C-8:	Sample Task Definition Form	C-30
Figure E-1:	Sendmail System Structure	E-6
Figure F-1:	Rewriting Set Semantics	F-37

Volume 2 — Devices and Networks

Figure 1-1:	A Disk Containing a 100% NCR UNIX Partition	1-8
Figure 1-2:	A Disk Containing Multiple Partitions	1-9
Figure 1-3:	The fdisk Utility Screen	1-11
Figure 1-4:	Appropriate Slice Assignments	1-15
Figure 1-5:	TAG Names and Descriptions	1-17
Figure 1-6:	Initially Installed Values for Swap Space	1-25
Figure 1-7:	Initial and Required Values for Dump Space	1-32
Figure 2-1:	Board Base Addresses and Interrupt Levels	2-21
Figure 3-1:	Example of sac, ttymon, login, and shell Relationships	3-10

Figures

Figure 4-1:	User Commands for the LP Print Service	4-8
Figure 4-2:	Administrator Commands for the LP Print Service	4-9
Figure 4-3:	Accepted content types	4-29
Figure 4-4:	Default Printer Port Characteristics ...	4-31
Figure 5-1:	Filter Keywords	5-39
Figure 5-2:	Information in Request Log (1 of 3) ..	5-47
Figure 5-3:	Information in Request Log (2 of 3) ..	5-48
Figure 5-4:	Information in Request Log (3 of 3) ..	5-49
Figure 6-1:	Non-PostScript Print Requests	6-4
Figure 6-2:	PostScript Translation Filters	6-7
Figure 6-3:	Special PostScript Filters	6-7
Figure 6-4:	Printer Port Characteristics	6-13
Figure 6-5:	Printer Definitions in terminfo (1 of 3)	6-15
Figure 6-6:	Printer Definitions in terminfo (2 of 3)	6-16
Figure 6-7:	Printer Definitions in terminfo (3 of 3)	6-17
Figure 6-8:	Exit Codes for Interface Programs	6-24
Figure 7-1:	Fields in netconfig Entries	7-3
Figure 7-2:	Sample netconfig File	7-8

	Figures
Figure 8-1: Format of the Command Log	8-16
Figure 8-2: Format of System History Log	8-17
Figure 8-3: Format of Error Log	8-18
Figure 8-4: Format of Transfer Log	8-19
Figure 8-5: Output for File Transfer Requests	8-20
Figure 8-6: Format of Accounting Log	8-21
Figure 8-7: Format of xfer Entry in Security Log . .	8-22
Figure 8-8: Format of rexe Entry in Security Log . .	8-23
Figure 8-9: Performance Log for conn Type Record	8-24
Figure 8-10: Performance Log for xfer Type Record	8-25
Figure 8-11: Format of Foreign Log	8-26
Figure 9-1: Modem Settings for BNU Features	9-7
Figure 9-2: BNU Escape Characters	9-17
Figure 9-3: Escape Characters in the Dialers File . .	9-32

Volume 3 — System Configuration

Figure 1-1: Cron Entries for sadc	1-6
Figure 1-2: sar Display Options (1 of 2)	1-9
Figure 1-3: sar Display Options (2 of 2)	1-10

Figures

Figure 1-4:	Example crontab Entries	1-37
Figure 1-5:	Example contrab Entry for Monacct	1-38
Figure 2-2:	How the Process Scheduler Works	2-4
Figure 2-3:	Scheduling Order and Global Priorities	2-9
Figure 3-1:	Configuration Files And Directories	3-3
Figure 3-2:	Options and Actions of the idtune Command	3-8
Figure 3-3:	Files in the Modules Directories	3-17
Figure 3-4:	Modules Available for the Kernel (1 of 7)	3-18
Figure 3-5:	Modules Available for the Kernel (2 of 7)	3-19
Figure 3-6:	Modules Available for the Kernel (3 of 7)	3-20
Figure 3-7:	Modules Available for the Kernel (4 of 7)	3-21
Figure 3-8:	Modules Available for the Kernel (5 of 7)	3-22
Figure 3-9:	Modules Available for the Kernel (6 of 7)	3-23
Figure 3-10:	Modules Available for the Kernel (7 of 7)	3-24
Figure 3-11:	General Kernel Parameters (1 of 2)	3-29

Figure 3-12:	General Kernel Parameters (2 of 2) . . .	3-30
Figure 3-13:	File System Parameters	3-31
Figure 3-14:	Paging Parameters	3-32
Figure 3-15:	pageout Daemon Parameters	3-32
Figure 3-16:	STREAMS Parameters	3-33
Figure 3-17:	log (strlog) Parameters	3-34
Figure 3-18:	IPC Message Parameters	3-35
Figure 3-19:	IPC Semaphore Parameters	3-35
Figure 3-20:	IPC Shared Memory Parameters	3-36
Figure 3-21:	RFS Parameters	3-37
Figure 3-22:	XENIX Parameters	3-38
Figure 3-23:	Miscellaneous Parameters	3-38
Figure 3-24:	Driver Parameters	3-39
Figure 3-25:	ASYNCIO Parameters	3-40
Figure 3-26:	EVENTS Parameters	3-41
Figure 3-27:	Timer and Scheduler Parameters	3-42
Figure 3-28:	Affinity Scheduling Parameters	3-42
Figure 3-29:	Resource Limit Parameters	3-43
Figure 3-30:	Overflow Buffers Parameters	3-44

Figures

Figure 3–31:	Shared Memory Nailing GIDs	3–44
Figure 3–32:	bdevcnt, cdevcnt Parameters	3–44
Figure 3–33:	Security Audit Trail Parameter	3–45
Figure 3–34:	Integrated LAN Driver Parameter	3–45
Figure 3–35:	Level 5 Tunable Parameters	3–45
Figure 3–36:	System Parameters	3–46
Figure 3–37:	Hardware Parameters	3–47
Figure 7–1:	Files and Directories That Grow	7–9

Common System Administration Tasks

Accessing the System	1-2
Changing Run Levels	1-5
Starting Up and Shutting Down the System	1-7
Specifying Run Levels During System Startup	1-9
Setting Up Your System for Automatic or Manual File System Checks	1-12
Setting Up an Alternate Boot Disk	1-17
Maintaining the Password and Shadow Files	1-28
Managing Software Packages	1-30
Managing Multiple Processors	1-32
Setting the Power Fail Strategy	1-37
Customizing Memory Dumps	1-44
Installing from a Remote Console (NCR 3450/3550 Only)	1-47

Accessing the System

A system administrator frequently uses any of the following logins:

- **sysadm** or **osa** logins
- **root** login
- **su** command

Using the **sysadm** and **osa** Logins

The **sysadm** login and the **osa** login both directly access the menu interfaces used to perform system administration tasks. You can also access these menus by logging in as **root** and then entering the **sysadm** command or **osa** command at the **root** prompt (**#**).

The system administrator menus provide the safest way to perform tasks on your system for two reasons:

- The menus warn you before or prevent you from removing important files.
- The menus automatically perform all steps in the procedure you select and verify the results.

Using the **root** and **su** Logins

There are a number of more advanced tasks that you can not perform through the menus. For these tasks you must use the **root** login or **su** login which permit you to access the system from the **root** prompt. Remember, the system assumes you know what you are doing when you are working from the **root** prompt and frequently does not warn you before performing commands that destroy disk information.

Permitting root Logins from Terminals

Sometimes you may want to log in as **root** from a terminal rather than from the system console. To permit non-console **root** logins, edit the */etc/default/login* file and either remove the following line or place a pound sign (#) in front of the line as shown:

```
# CONSOLE=/dev/console
```

Enabling non-console **root** logins makes the system less secure. Instead, the system administrator may login remotely as a regular user and then use the **su(1M)** command to access **root** privileges.

Using Virtual Terminals for Multiple Console Logins

Virtual terminals permit you to use multiple login screens on the console or monitor and to switch back and forth between them. Using virtual terminals, you can do several things almost simultaneously (for example, edit a file and check the status of running processes).

Setting Up Virtual Terminals

To set up virtual terminals, use either the **vtlmgrr(1M)** or **newvt(1M)** command.

- **vtlmgrr** opens a virtual terminal when you try to access one.
- **newvt** automatically opens the next available virtual terminal and switches you to that virtual terminal.

Accessing Virtual Terminals

To access a virtual terminal, press the ALT–PRTSC keys simultaneously. Then press the function key that corresponds to the number of the virtual terminal you want to use. For example, to access the first virtual terminal (**vt01**), press ALT–PRTSC and then press the F1 function key.

To return to */dev/console*, press the ALT–PRTSC keys and then the letter **h**.

Closing Virtual Terminals

Enter CTRL-D to log off the system console. You are asked if you want to close all virtual terminals.

- Enter the letter **y** and press RETURN to close all virtual terminals and log off the system.
- Enter the letter **n** if you want to return to the virtual terminals before logging off.

Changing Run Levels

You can change run levels (system states) using the **shutdown(1M)** or **init(1M)** command.

- You should use the **shutdown(1M)** command to change to run levels 0, s, 1, or 6. See the “Performing a System Shutdown” section later in this chapter for instructions on using **shutdown(1M)**.
- You must use the **init(1M)** command to change to run levels. Use **init(1M)** as follows:

```
# init run_level
```

Figure 1–1 and Figure 1–2 describe the possible run levels.

Run Level	Description
0	Power-down state. Shuts the machine down so you can safely remove the power supply.
s	Single user state. This run level should be used when installing or removing software utilities, checking file systems, or using the Maintenance (<i>/install</i>) file system. It is similar to run level 1; however, in run level s, multi-user file systems are unmounted and daemons are stopped.
1	Administrative state. In run level 1, file systems required for multi-user operations are mounted, and logins requiring access to multi-user file systems can be used. No services are stopped when entering run level 1 from run level 2.
2	Multi-user state. File systems are mounted and normal user services are started.
3	Network File System (NFS) state. Prepares your system to use NFS.

Figure 1–1: Run Levels (1 of 2)

Common System Administration Tasks

Run Level	Description
4	User-defined.
5	Firmware state.
6	Power-down and reboot to the state defined by the initdefault entry in the <i>/etc/inittab</i> file.

Figure 1–2: Run Levels (2 of 2)

Starting Up and Shutting Down the System

Performing a System Startup

You can cause a system startup by pressing the power switch to turn the power off and pressing it again to begin the startup.

Warning

Unless your system can perform a powerfail recovery in case of power loss (like the NCR 355x and 3450 with UPS), shutting down the system by turning off the power switch causes file system corruption.

For 33xx systems, you can perform a system startup by simultaneously pressing the CTRL-ALT-DEL keys if the system is down. Pressing the power switch to turn the system off causes system memory to be cleared whereas pressing the CTRL-ALT-DEL keys does not.

Performing a System Shutdown

To perform an orderly shutdown, use the `shutdown(1M)` command.

You must be in the *root* directory to perform a shutdown.

- To shutdown the system (immediately) and reboot, enter the following:

```
# cd /  
# shutdown -i6 -g0 -y
```


Common System Administration Tasks

- To shutdown the system, starting in *X* seconds and NOT reboot, enter the following:

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

NOTE: If you do not specify *gX*, the default grace period before shutdown begins is 60 seconds.

Specifying Run Levels During System Startup

Beginning with Release 2.02 of the NCR UNIX SVR4 MP-RAS operating system, you can configure your system so that you can specify the desired run level during system startup.

Configuring Your System

If you want to be able to specify the desired run level during system startup, perform the following steps:

Step 1: Create the following script with the name */sbin/ask_level*:

```
INITTAB=/etc/inittab      # Init script
TIMEOUT=10                # Seconds allowed for user to
                           # decide on run level

#
# Search inittab for initdefault entry and echo to
# standard output
#
GetInitDefault()
{
    [-r $INITTAB] || return
    OLD_IFS="$IFS"         # save old field separator
    IFS=":" export IFS     # inittab separator is :

    while read ID RSTATE ACTION PROCESS
    do
        if ["$ACTION"="initdefault" -a -n "$RSTATE"]
        then
            echo $RSTATE
            return
        fi
    done
    IFS="$OLD_IFS"
}
```

Common System Administration Tasks

```
INIT_DEFAULT='GetInitDefault'
[-n "$INIT_DEFAULT"] && echo "Default run level is
    $INIT_DEFAULT"

# Offer user opportunity to select run level, but
# don't wait forever!
if readtimeout $TIMEOUT "Select a different run level"
then
    while [ 0 -eq 0 ]
    do
        echo "Enter preferred run level (0-6,s): \c"
        read ANS
        case $ANS in
            "" )           exit 0 ;; # take default
            0 )           exit 8 ;;
            1|2|3|4|5|6)   exit $ANS ;;
            s|S )         exit 7 ;;
            * )           echo "\"$ANS\" is not a valid
run level ;;
            esac
        done
    fi
exit 0
```

If you modify this script, be sure not to use any commands other than those on your root file system. For example, don't use commands under */usr/sbin* if */usr* is a mounted file system.

Typical permissions for */sbin/ask_level* are as follows:

```
-r-xr-xr-x 1 root sys 1374 Jul 2 10:26 /sbin/ask_level
```

Step 2: Create an entry similar to the following in */etc/inittab*. Place this entry anywhere before the *initdefault* entry.

```
ask::sysconf:/sbin/ask_level </dev/console >/dev/sysmsg 2>&1
```

Step 3: Add the same entry to your */etc/conf/df.d/init.base* file.

Setting the Run Level

During system startup, *ask_level* displays the default run level from your */etc/inittab* file and asks if you want to select a different run level.

- Step 1:** If you do not want to specify a different run level, enter `n`. If you do want to specify a different run level, enter `y`; the script then prompts you for your preferred run level.
- Step 2:** Enter the desired run level (0 through 6 or `s`). You must respond within a limited amount of time (10 seconds by default). If you do not respond within that time, the system boots to the default run level, as specified in */etc/inittab*.

Setting Up Your System for Automatic or Manual File System Checks

When your system goes into multiuser mode, the software checks for file system corruption. You can have your system check and repair the file systems automatically or prompt you to interactively fix any problem.

Automatic Checking

If you choose to have the **fsck** utility run automatically on every file system as part of the normal start-up procedures, **fsck** will be performed using the option **-y**, which means that **fsck** responds yes to all questions trying to fix a corrupted file system. Although the system will be able to start up stand-alone, there is a possibility of losing data unnecessarily. Orphaned files and directories (allocated but not referenced) are put into the *lost+found* directory.

Manual Checking

If you choose to run **fsck** manually, you will be prompted to invoke **fsck** and fix all problems interactively in case the system finds corrupted file systems at start-up time. Although the system will not be able to startup stand-alone in case of corruption, you will be able to respond to each problem **fsck** finds individually.

Specifying Type of Checking

Whether your system performs an automatic file system check on file systems or prompts you to check the file systems manually in case of corruption depends on the following:

- The configuration of the file */etc/vfstab*
- The installation of the **support** software package. The **support** package is distributed with the base operating system package tape and can be installed to enable manual file system checking for all file systems including the root file system.

To check whether the **support** package is installed, perform the following command:

```
# pkginfo support
```

If this command reports the **support** package is installed, you should also find out whether automatic or manual file system checking was specified during installation. The suggested default is to use automatic file system checking. If the file */etc.NCRM* exists, then you specified to use manual file system checking at the installation time of the **support** package. (Note that prior to release 2.00.02, the file */etc.NCRM* was created with the installation of the package named **ncrm**. However, for release 2.00.02 and greater, this file no longer has anything to do with the package named **ncrm**). This file does not exist if the **support** package was installed using the default of automatic file system checking.

When the System Performs an Automatic File System Check

The file systems will be checked and repaired automatically if one of the following is true:

- The **support** package is not installed and the field *fsckpass* in the file */etc/vfstab* is set to **1** specifying an automatic file system check. For detailed information on the */etc/vfstab* file, refer to “General Administration of File Systems” in Chapter 4.

- The **support** package is installed and you specified to use automatic file system check at system startup time and the value of the field *fsckpass* in the file */etc/vfstab* is set to a 1 specifying an automatic file system check.
- Starting with Release 2.02 of NCR UNIX SVR4 MP-RAS, if either *fsckdev* or *fsckpass* fields contain a dash (–) then automatic file system repair is turned off. If the *fsckdev* field contains a device and the *fsckpass* field contains a dash, the device is checked for mountability before mounting; however, it is not repaired if errors exist. A WARNING statement is displayed during boot, indicating that an error was found and must be repaired before the device can be mounted. However, the root file system is always checked automatically unless the **support** package is installed with the manual file system check option.

NOTE: By default, the field *fsckpass* contains a 1, indicating to perform an automatic file system check.

When the System Prompts for a Manual File System Check

The root file system will NOT be automatically checked if the **support** package is installed and you specified to use manual file system check at system startup time. In this case, the **support** package will check all file systems listed in the file */etc/vfstab* with a *fsckpass* field of 1 for possible corruption. This configuration allows for manual file system checks for all the file systems, including the root file systems. If corruption is present for a file system, you are asked to manually run the file system check.

The value in *fsckpass* for the root file system and */usr* is ignored. Since the root and */usr* file systems are important for the sane operation of the system, these file systems are always checked for corruption.

NOTE: Do NOT change the field *fsckpass* to a – for the root file system if you have the **support** package installed with the manual file system check option. If you change the field *fsckpass* to a “–” for any non-root file system, they will be checked automatically.

If you have the **support** package installed for manual checking and if corruption is detected, the system will always prompt the system administrator to run **fsck** manually with the message similar to the following:

```
The system cannot go multi-user until the following
file systems are checked:
```

Partition	Mount point
-----	-----
/	/dev/rroot
/usr	/dev/rdisk/c0t6d0s3

The root file system should be checked as follows:

1. If file systems other than root need checking, check them first.
2. Execute 'uadmin 4 128'
3. Check the root file system.
4. Execute 'uadmin 1 2' to reboot the system with no superblock update.

If this message appears, do the following:

Step 1: Use **fsck** to check and repair any non-root file systems listed in the display. For example:

```
# fsck -y /dev/rdisk/c0t6d0s3
```

See Chapter 5, “Checking and Repairing the File System,” for more information on using **fsck**.

Step 2: Enter the following command:

```
# uadmin 4 128
```


Common System Administration Tasks

Step 3: Use **fsck** to check and repair the root file system. For example:

```
# fsck -y /dev/rroot
```

See Chapter 5, “Checking and Repairing the File System,” for more information on using **fsck**.

Step 4: Enter the following command to reboot the system:

```
# uadmin 1 2
```

Setting Up an Alternate Boot Disk

Why an Alternate Boot Disk?

An alternate boot disk is useful when you want to minimize the downtime of the system in the event of either of the following:

- Software failure
- Hardware failure

You may set up an alternate boot disk in systems with root mirroring, but doing so requires an additional disk. While root mirroring makes your system media tolerant by having a working backup copy of the original root disk available at any given time, it does not protect the system against software failures.

Also, in case of disk failure with root mirroring, the mirrored disk must be physically moved to the original boot disk slot. With root mirroring, you can avoid system downtime only if Hardware Manager is installed on your system. Hardware Manager allows you to hot-plug the failed disk. Otherwise, you must physically move the mirror boot disk to the original boot disk slot for NCR 35xx systems or re-strap the mirror disk to the original disk device ID for NCR 345x and uniprocessor systems. It is necessary to move or re-strap the disk because the root mirror is an exact copy of the original root disk; therefore, devicenames, major numbers, and minor numbers under the */dev* directory are the same, as well as the mount points in the */etc/vfstab* file.

Configuring the Alternate Boot Disk

In the following procedure for configuring the alternate boot disk, the example primary boot disk is `/dev/dsk/c0t6d0s0` and the example alternate boot disk is `/dev/dsk/c0t5d0s0`.

- Step 1: Format, partition and slice the alternate boot disk to be identical to the primary boot disk. When slicing the disk, the device nodes for the slices and the tags identifying the slice type are set up automatically when using the `sysadm` or OSA system administration menus.

If you use the command line, use the following commands:

- `dkformat` to format the disk
- `fdisk` to partition the disk
- `disksetup` to slice the disk

Also run `mktable` after adding or slicing a disk from the command line.

- Step 2: Make sure that the volume table of contents structure (`vtoc`) on the alternate boot disk contains matching tags for slice 1, slice 2, slice 10 (slice a) and slice 7. The tag is the identifier for the type of slice; for example, slice 1 is ROOT, slice 2 is SWAP, slice 7 is BOOT and slice 10 is STAND. The tag definitions are found in `/usr/include/sys/vtoc.h`.

When you use the `prtvtoc` command and save the output to a file, the tags are given in hexadecimal form. For the ROOT slice, the tag is 0x2; for the SWAP slice, the tag is 0x3; for the BOOT slice the tag is 0x1; and for slice 10 the tag is 0x9. If you have `/usr` or `/var` on separate file systems or if you want any user-defined file systems such as `/home` also on the alternate boot disk, be sure to correct these tags as well.

To correct these tags, enter the following:

```
# prtvtoc -f filename1 alternate_boot_device
# prtvtoc -f filename2 primary_boot_device
```

For example:

```
# prtvtoc -f /tmp/disk5 /dev/rdisk/c0t5d0s0
# prtvtoc -f /tmp/disk6 /dev/rdisk/c0t6d0s0
```

Compare the tags in the vtoc structure saved for the primary boot device with the tags saved for the alternate boot device. Edit the saved vtoc structure of the alternate boot device and make the tags the same as for the primary boot device, using the following command:

```
# vi /tmp/disk5
```

Once the tags in the vtoc structure of the alternate boot device are the same for the primary boot device, use the following command to copy the updated vtoc structure to the alternate boot device:

```
# edvtoc -f /tmp/disk5 /dev/rdisk/c0t5d0s0
```

You will be prompted to answer **y** or **n** to overwrite the vtoc structure on disk. Answer **y**.

Step 3: Write the boot program into blocks 1–28 of the *BOOT* slice (7):

```
# dklayout -b -f /etc/boot -d /dev/rdisk/c0t5d0s0
```

NOTE: You have to specify slice 0 (whole disk) as a device for the **-d** option.

Step 4: Put the system in single user mode:

```
# init s
```

Common System Administration Tasks

Step 5: Copy the data from the primary boot disk to the alternate boot disk.

If the primary and alternate boot disks are identical and are sliced identically, you can copy the whole disk:

```
# dd if=/dev/rdisk/c0t6d0s0 bs=200k \  
of=/dev/rdisk/c0t5d0s0
```

Note that slice 0 does not include any other operating systems partition such as a DOS partition. If you have a DOS partition, you must copy slice 5 separately:

```
# dd if=/dev/rdisk/c0t6d0s5 bs=200k \  
of=/dev/rdisk/c0t5d0s5
```

If the primary and alternate boot disk are not identical, or if you have two disks of different capacities, you must copy each slice separately. The slices on the alternate boot disk must be at least equal in size or bigger than the slices on the original disk.

```
# dd if=/dev/rdisk/c0t6d0s1 bs=200k \  
of=/dev/rdisk/c0t5d0s1
```

```
# dd if=/dev/rdisk/c0t6d0sa bs=200k \  
of=/dev/rdisk/c0t5d0sa
```

```
# dd if=/dev/rdisk/c0t6d0sX bs=200k \  
of=/dev/rdisk/c0t5d0sX
```

where *X* is the slice number of any other mounted file system on the root disk such as */usr*, or */var*, or user file systems such as */home*.

Step 6: Check the file systems on the alternate boot device:

For the *root* file system, enter:

```
# fsck -F ufs /dev/rdisk/c0t5d0s1 OR  
# fsck -F s5 /dev/rdisk/c0t5d0s1
```

For the *stand* file system, enter:

```
# fsck -F bfs /dev/rdsk/c0t5d0sa
```

For any other file systems, such as */usr*, on slice 3, enter

```
# fsck -F FStype /dev/rdsk/c0t5d0s3
```

Step 7: Note the major and minor numbers for the root and swap slice of the alternate boot disk:

```
# ls -l /dev/dsk/c0t5d0s1
# ls -l /dev/dsk/c0t5d0s2
```

Step 8: Create a */stand/boot* file for the alternate boot disk:

```
# cp /stand/boot /stand/boot2
```

Edit */stand/boot2* so that it contains the following entries:

```
rootdev=SCSI (X, 0)
swapdev=SCSI (Y, 0)
```

where *X* is the minor number for the root slice (slice 1) of the alternate boot disk and *Y* is the minor number for the swap slice (slice 2) of the alternate boot disk.

NOTE: By adding these appropriate entries for the root slice and swap slice in the */stand/boot* file, a kernel rebuild is avoided because the entries in this file override defaults compiled into the kernel.

If there is a disk array on the machine, and the disk array software is installed (*rdac* software package which is dependent on the *dau* software package for RAID), use the following entries in */stand/boot2* instead:

```
rootdev=rdac (X, 0)
swapdev=rdac (Y, 0)
```

You must do this EVEN though the boot and alternate boot disks are not part of the disk array.

Common System Administration Tasks

Step 9: Create an appropriate */etc/vfstab* file for the alternate boot disk:

```
# cp /etc/vfstab /etc/vfstab2
```

Correct all entries for any file system mounted on the root disk. For example:

```
/dev/dsk/c0t6d0sa /dev/rdsk/c0t6d0sa /stand  
bfs 1 yes -
```

must be changed to

```
/dev/dsk/c0t5d0sa /dev/rdsk/c0t5d0sa /stand  
bfs 1 yes -
```

Step 10: Copy and edit the file */etc/swapnodes* if:

a) Your system has secondary swap space residing on the original disk, and you do not want to swap to the original boot disk after booting from the alternate boot disk.

b) You do not want to depend on the original root disk for secondary swap space if the original root disk has a hardware failure.

If you need to change the */etc/swapnodes* file, make a copy and edit the copy:

```
# cp /etc/swapnodes /etc/swapnodes2
```

Change all device names referring to the original root disk to corresponding device names on the alternate disk.

- Step 11: Mount the *root* file system of the alternate boot disk. Make sure your */mnt* directory is empty and not already serving as a mount point for a file system. Copy the edited copy of the */stand/vfstab* file to the alternate boot disk:

```
# mount -F FSType /dev/dsk/c0t5d0s1 /mnt
# cp /etc/vfstab2 /mnt/etc/vfstab
```

If you created the file */etc/swapnodes2*, enter:

```
# cp /etc/swapnodes2 /mnt/etc/swapnodes
```

- Step 12: Mount the *stand* file systems of the alternate boot disk. Copy the edited copy of the */stand/boot* file to the alternate boot disk:

```
# mount -F bfs /dev/dsk/c0t5d0sa /mnt/stand
# cp /stand/boot2 /mnt/stand/boot
```

If you edited the file */etc/swapnodes*, enter:

```
# cp /etc/swapnodes /mnt/etc/swapnodes
```

- Step 13: Create the correct block and character device nodes for *root* and *rroot* on the alternate boot disk:

```
# cd /mnt/dev
# rm root rroot
# ln /mnt/dev/dsk/c0t5d0s1 /mnt/dev/root
# ln /mnt/dev/rdsk/c0t5d0s1 /mnt/dev/rroot
```

- Step 14: If the swap slice resides on the root disk, create the correct block and character device nodes for *swap* and *rswap* on the alternate boot disk:

```
# cd /mnt/dev
# rm swap rswap
# ln /mnt/dev/dsk/c0t5d0s2 /mnt/dev/swap
# ln /mnt/dev/rdsk/c0t5d0s2 /mnt/dev/rswap
```


Step 15: Unmount the file systems of the alternate boot disk:

```
# cd /  
# umount /mnt/stand  
# umount /mnt
```

Step 16: If you have a uniprocessor system, perform the steps in the procedure “Bootting from the Alternate Boot Disk on Uniprocessor Systems.” If you have a multiprocessor system, perform the steps in the procedure “Bootting from the Alternate Boot Disk on Multiprocessor Systems.”

Step 17: Repeat steps 4 through 6, 9 through 11, and 15 after data or configuration changes are made to the primary boot disk to ensure the alternate boot disk contains the most recent data. You can easily write a shell script to perform these steps automatically and then use the `cron` command to have it execute automatically at certain intervals, such as every week.

Steps 1 through 3, 7, 8, and 12 through 14 need only be performed once.

NOTE: Execute step 17 always before and after updating or installing software.

Bootting from the Alternate Boot Disk on Uniprocessor Systems

On uniprocessor systems, you can only boot from devices strapped to SCSI ID 5 or SCSI ID 6. If the device is strapped to SCSI ID 6 is not accessible, the system will boot from the device strapped to SCSI ID 5. If the device strapped to SCSI ID 6 is accessible, the system will attempt to boot from that device.

To automatically boot from the alternate boot disk on uniprocessor systems, do the following:

Step 1: Perform a system shutdown, by entering:

```
# shutdown -y -i0 -g0
```

- Step 2: Power off the system.
- Step 3: If the primary boot device is an external device, power it off , but do not disconnect it from the SCSI bus. Once the system is booted, the primary device can be powered on and accessed.
- Step 4: If the primary boot device is an internal device, physically remove it from the cabinet.
- Step 5: Start up system.
- Step 6: Update the dump device information. If you plan to remove the original root disk once you boot off the alternate boot disk or if you don't want the system to dump to the original root disk, you should delete any dump space allocated on the original root disk and add the dump space again on the alternate boot disk.

Use the following command to write the CMOS dump table to the */etc/dumpnodes* file:

```
# fdump -w
```

Edit */etc/dumpnodes*, changing dump entries on the original root disk to the alternate boot disk. Then enter the following command to write the entries from */etc/dumpnodes* to the CMOS dump table:

```
# fdump -i
```

Booting from the Alternate Boot Disk on Multiprocessor Systems

In the following procedure for booting from the alternate boot disk, the example primary boot disk is `/dev/dsk/c0t6d0s0` and the example alternate boot disk is `/dev/dsk/c0t5d0s0`. To automatically boot from the alternate boot disk on multiprocessor systems, do the following:

Step 1: Set up the primary and alternate boot device using the **setboot** command:

```
# setboot -p -u /dev/rdisk/c0t6d0s0
# setboot -s -u /dev/rdisk/c0t5d0s0
```

You can also go into the Startup Subsystem Services (SUS) menus and set the primary and secondary boot device by selecting **System Boot Services** from the SUS main menu.

Should the primary boot device not be a bootable device, the system will automatically boot from the secondary boot device.

Step 2: Test your alternate boot device or set the alternate boot device to be your primary boot device, using the following command:

```
# setboot -p -u /dev/rdisk/c0t5d0s0
```

Step 3: Reboot the system:

```
# shutdown -y -i6 -g0
```

Step 4: Update the dump device information. If you plan to remove the original root disk once you boot off the alternate boot disk or if you don't want the system to dump to the original root disk, you should delete any dump space allocated on the original root disk and add the dump space again on the alternate boot disk.

Use the following command to write the CMOS dump table to the `/etc/dumpnodes` file:

```
# fdump -w
```

Edit */etc/dumpnodes*, changing dump entries on the original root disk to the alternate boot disk. Then enter the following command to write the entries from */etc/dumpnodes* to the CMOS dump table:

```
# fdump -i
```

Step 5: After you have tested the alternate boot disk, set the primary boot disk to the original boot disk:

```
# setboot -p -u /dev/rdsk/c0t6d0s0
```

Step 6: List the device names for the primary and secondary boot disks. The following command lists the primary boot device:

```
# setboot -p -l /dev/rdsk/c0t6d0s0
```

The following command lists the secondary boot device:

```
# setboot -s -l /dev/rdsk/c0t6d0s0
```

NOTE: You may specify ANY disk device name that exists on your system; it need not be the primary boot disk.

Maintaining the Password and Shadow Files

What Is Password Shadowing?

Password shadowing is a security enhancement that is required in NCR UNIX SVR4. Before password shadowing, encrypted passwords were stored in the password file, which was readable by all users. Unfriendly users (remote and local) could examine the password file to find logins that had no password and could then use these logins to gain access to the system.

To deter such activities, encrypted passwords and their attributes, such as aging information, have been moved to an access-restricted file called the shadow password file (*/etc/shadow*). This file is readable only by its owner and root. The field in the password file (*/etc/passwd*) that formerly contained the encrypted password now normally contains the letter x.

Relationship Between */etc/passwd* and */etc/shadow*

For a login to be used, it must have a one-line entry in both the */etc/passwd* and */etc/shadow* files. The first field in the entry is the login name; the second field is for the password. In the */etc/passwd* file, the password field normally contains the letter x. In the */etc/shadow* file, the password field contains the encrypted password. Sample entries for the root login are:

- */etc/passwd:* root:x:0:1:0000-Admin(0000):/:
- */etc/shadow:* root:LSg9PiFXIX562:7657:0:168:7:::

Figure 1–3 shows the relationship between the password fields in */etc/passwd* and */etc/shadow*:

Password Field Entry		Possible Login Activity
<i>/etc/passwd</i>	<i>/etc/shadow</i>	
x	encrypted password	Login permitted; you must specify the correct password
x	no entry	Login permitted; no password required
x	x	No login permitted (locked login)
no entry	encrypted password	No login permitted
no entry	no entry	Login permitted; you must specify a new password when you log in
no entry	x	No login permitted (locked login)

Figure 1–3: Relationships Between */etc/passwd* and */etc/shadow*

Updating the */etc/shadow* File

You may use the following commands to re-create or update the */etc/shadow* file:

- The **pwconv(1M)** command creates or updates the */etc/shadow* file from entries in */etc/passwd*.
- The **passmgmt(1M)** command adds, deletes, or changes information for a given user in the */etc/shadow* file.

Managing Software Packages

From time to time, you may want to install or remove driver or other software packages. You may also want to determine what driver or software packages are installed on your system. The following procedures describe how to list, install, and remove packages.

Listing Software Packages

To list all driver or software packages that are installed on your system, enter the following:

```
# pkginfo
```

Installing Software Packages from Flex Diskette

To install all driver or software packages from flex diskette, place the diskette in the flex drive and enter the following:

```
# pkgadd -d /dev/dsk/device_name
```

See Appendix B, “Device Names and Numbers,” for an explanation of device names.

Installing Software Packages from Cartridge Tape

To install all driver or software packages from cartridge tape, place the tape in the tape drive and enter the following:

```
# pkgadd -d /dev/rmt/device_name
```

See Appendix B, “Device Names and Numbers,” for an explanation of device names.

Removing Software Packages

To remove a driver or software package that is installed on your system, enter the following:

```
# pkgrm package_name
```


Managing Multiple Processors

Several commands are available for managing the processors on multiprocessor systems. These commands allow you to perform the following tasks:

- Display the status of processors (active or inactive)
- Activate or de-activate processors using a shell command
- Bind a process to a processor using a shell command
- Display process binding information

Display the Status of Processors

The **pinfo** command displays the status of processors in the system. The status is one of the following:

- **online** — indicates that a processor is active and capable of executing processes
- **offline** — indicates that a processor has been logically removed from the system

To display the status of all processors on your system, enter the following command:

```
# pinfo
```

```
processor 0: online
processor 1: online
processor 2: offline
processor 3: offline
```

To display the status, processor type, clock speed, and floating point unit type for a specific processor, use the **-v** option and the number of the processor. For example, to display this information for processor 0, enter the following command:

```
# pinfo -v 0
```

```
processor 0: online
      cpu P_486, 50 MHz, fpu P_87
```

Activate or De-activate Processors

The **online** and **offline** commands change the status of all or specified processors to online (active) or offline (inactive).

- **online** — indicates that a processor is active and capable of executing processes
- **offline** — indicates that a processor has been logically removed from the system

NOTE: Use the **offline** command with caution, particularly on a loaded or stressed system.

The online Command

To change the status of all processors to online, enter the following command:

```
# online
```

To change the status of a specific processor, use the number of the processor. For example, the following command changes the status of processor 1 to online:

```
# online 1
```

Using the **-v** option displays the status of the processor before and after the **online** command was issued. For example:

```
# online -v 1
```

```
processor 1: off -> on
```

The offline Command

To change the status of all processors to offline, enter the following command:

```
# offline
```

To change the status of a specific processor, use the number of the processor. For example, the following command changes the status of processor 1 to offline:

```
# offline 1
```

Using the **-v** option displays the status of the processor before and after the **offline** command was issued. For example:

```
# offline -v 1
```

```
processor 1: on -> off
```

NOTE: You can not de-activate the boot processor. If you attempt to de-activate the boot processor or a processor that is executing bound processes, you receive the following error message:

```
processor 0: Device busy
```

Bind a Process

The **pbind** command allows you to bind processes to a specific processor. You can also use this command to unbind a process or to display binding information for specific process IDs.

The pbind Command

The **pbind** command binds processes to a specific processor. For example, the following command binds process 163 to processor 1:

```
# pbind 1 163
```

Specifying a dash (–) removes binding for the specified process(es). For example, the following command removes the binding from process 163:

```
# pbind - 163
```

The **-q** option displays binding information for the specified process or processes. If you do not specify processes, binding information for the entire system is displayed.

Display Process Binding Information

Using the **-P** option with other commands displays information about bound processes.

ps -P

The **-P** option of the **ps** command displays the processor ID of the processor on which a bound process is executing. This option may also be used with other **ps** command options.

```
# ps -P
```

PID	PSR	TTY	TIME	COMD
163	-	console	13:43	sh
251	-	console	14:12	ps

The **-P** option adds the **PSR** column to the output. Neither the shell nor the **ps** command are bound to a specific processor, so the field contains a dash.

sar -P

The **sar -P** command causes the other specified options to display the requested information about a specific processor, rather than the entire system. The **-P** option may be used with the following options: **-a, -b, -c, -g, -m, -u, -w, -y, -D**.

For example, the following command displays the processor utilization for processor 0 over a 5 second interval.

```
# sar -u -P 0 5
```

```
UNIXV.4 liz 4.0 3.0 3550 Processor 0 09/27/91
13:43:04 %usr %sys %wio %idle
13:43:09 9    7    4    80
```

prfpr -P

The **prfpr -P** command displays the profile information on a per-processor basis, rather than on a system-wide basis. For example, the following command displays the profile information from file *profile_file* on a per-processor basis.

```
# prfpr -P profile_file
```

Setting the Power Fail Strategy

There are two possible strategies that the system can use in case of power loss:

- **Orderly Shutdown**

Orderly shutdown means the system will detect the power failure, halt all activity and shut the system down in an orderly manner. Users must log in again, and restart all applications.

- **Powerfail Recovery**

Powerfail Recovery means the system will detect the power failure, write the contents of memory to the DUMP partition on disk. Upon “power up,” the system will boot UNIX, restore memory from the dump partition, and start up all processes that were running at the time of the power failure. The system comes back to where it was at the time of failure. Some applications, such as networking and X windows, are restarted.

Requirements for Multiprocessor Systems

The following are necessary requirements for Powerfail Recovery in multiprocessor systems:

- You must have the optional Power Backup System (PBS) module installed. The battery power for the NCR 355x PBS comes from four batteries located in the bottom of the cabinet. The battery power for the NCR 345x PBS comes from a removable container in the rear of the cabinet.

- The Power Backup System requires that the CMOS battery be present and
 - For a NCR NCR 345x, the CMOS battery, located on the bottom of the front of the unit, must be turned on.
 - For a NCR 355x system, the CMOS battery, located at the top of the unit, must be fused.
- Your system must have a system dump area configured. At system installation time, slice 6 is the default to be reserved for the dump area and is set up to a size equal to the memory for some memory configurations. However, the dump area is not restricted to physically residing on the root disk. The total dump space allocated must be equal to the total amount of physical system memory available, but dump space can be distributed among different disk drives.
- On large memory configurations, dump space must be distributed across multiple SCSI busses in parallel to conserve battery power.

Requirements for Uniprocessor Systems

The following are necessary requirements for Powerfail Recovery in uniprocessor systems:

- You must have the optional free standing power supply unit 4098 UPS (Uninterruptible Power Supply) installed as well as the UPS software package. (The NCR 3447 has an internal power backup system).
- The serial interface to the UPS must be via the second serial tty port (except on an NCR 3447).
- The system must have a RBIOS (ROM-Based Remote Diagnostics) version of 1.1 or later to support SCSI BUS reset.

- If the flex drive is in use at the time of power loss, the floppy must be removed prior to the system recovery.
- If a mouse is to be used over a serial interface, the mouse port can not be port 0 or 1 (except on an NCR 3447).

Caution

If the powerfail strategy is set to powerfail recovery and the system has no dump area configured or the dump area is not large enough to hold the memory, the system will automatically switch the strategy from powerfail recovery to orderly shutdown.

The following chart shows the Power Backup System hardware supported for the System 3000 and the software packages required:

NCR SYSTEM	Power Backup System Hardware Supported	Minimal O.S. Release	Required Software
355x	Internal	2.00.01	none
345x	Internal	2.00.01	none
3447	Internal	2.00.02	UPS
3447	External (4098)	1.00.01	UPS
3445	External (4098)	1.00.01	UPS
All Level 3 Uniprocessors	External (4098)	1.00.01	UPS

Figure 1-4: Power Backup System Support Hardware

NOTE: The UPS Package Release 2.00.03 supports both powerfail recovery strategies for orderly shutdown and powerfail recovery.

Customizing Powerfail Strategy

You can choose to set the default powerfail strategy to:

- Powerfail recovery
- Shutdown

For Release 2.01, you can also set the default strategy through the Open System Administrator (OSA) menus. When using the command line, use the new **strategy** command to set the powerfail strategy permanently. The **uadmin** command sets the powerfail strategy only temporarily.

Displaying the Current Powerfail Strategy

The following command displays the current powerfail recovery strategy:

```
# strategy
```

Setting Powerfail Strategy to Shutdown

You can choose to change the powerfail strategy to shutdown. Use the following command:

```
# strategy -s
```

Setting your powerfail strategy to shutdown causes the system not to perform a powerfail recovery; the system performs an orderly shutdown instead.

Setting Powerfail Strategy to Powerfail Recovery

Powerfail recovery is the default strategy for powerfail. If you changed it to shutdown and want to set it back to powerfail recovery, use the command:

```
# strategy -p
```

NOTE: When the powerfail strategy is set to powerfail recovery and the system loses power upon recovery, the system always tries to load the contents of the dump area back into memory. You cannot interrupt the powerfail recovery process to boot from hard disk.

The **strategy** command will use the */etc/default/dump* file to determine the current strategy or to set up a different strategy.

Error Reporting

If the dump area is setup incorrectly, the system automatically sets the powerfail strategy to orderly shutdown. Each time the system performs a startup, the script in */etc/init.d/DUMPCHK* is executed. This script checks the following:

- Overlapping of swap and dump areas
- Size of dump space less than memory size
- SCSI bus distribution

For systems with more than 512 MB of memory, the dump area must be spread across available disks and system busses so that no more than 512 MB of memory is dumped per SCSI bus.

If an error is encountered, it is reported to all users logged in as root and an entry in the error log file is made. If the strategy was set to powerfail recovery, it is automatically changed to shutdown.

The */etc/init.d/DUMPCHK* script is executed through cron every hour to check the dump space; errors are reported on the console as well as logged until the dump space is set up correctly.

What Happens at Powerfail Recovery?

The Power Backup System (PBS) provides battery supplied power necessary to perform the specified powerfail strategy in the event of a power failure. For power failures of 10 seconds or less on multiprocessor systems, power from the PBS simply protects the system data and the memory is not dumped.

Strategy Set to Shutdown

Once the shutdown starts, it runs to completion, even if power is restored during shutdown.

Strategy Set to Recovery

- The system batteries maintain the power supply during the memory dump.
- If power is restored during the dump, the system returns to normal operation.
- Once the memory is dumped, the power supply is turned off to conserve the system batteries until power is restored.
- Once the power is restored, the system automatically reboots and loads the dumped memory contents back into memory.

When power is restored, the system is brought back to the same state it was in the time of power failure. For example, most applications and processes return to the point of execution before the power failure occurred. Applications requiring networking or X Windows must be restarted.

The PBS can provide enough power for a minimum of 10 minutes. Information messages are displayed on the console at the time of power failure.

The batteries require 8 hours to recharge to 75% from a fully discharged state. Depending on the duration of the power loss, recharging the batteries can take from several minutes to hours. During startup, the Startup Subsystem (SUS) determines the state of the batteries. By default, the system automatically boots. Be aware that the system might have batteries that have not been fully recharged. If you have the Open System Administrator (OSA) package installed, you can query the state of the batteries from the Hardware Manager.

Powerfail Recovery in a Network Environment

Normally, power failure causes all communications to be dropped. For security reasons, all network daemons and processes, database daemons and processes, etc., are shut down and restarted by the *init* process after power fail recovery:

- The *init* process reads the */etc/inittab* file to find the network powerfail routine */sbin/rcpfr*.
- The *init* process executes the */sbin/rcpfr* script.
- This */sbin/rcpfr* script executes the scripts in the */etc/rcp?.d* directory, where ? corresponds to the current run level. These scripts are executed in alphabetical order starting with the shutdown scripts and then the startup scripts. Each script is run with the appropriate start/stop argument.

NOTE: If you have written your own network applications, remember to link the scripts in the */etc/init.d* directory to the */etc/rcp?.d* directory.

Example

The RPC package links the */etc/init.d/rpc* script to */etc/rcp2.d/K25rpc*, */etc/rcp2.d/S75rpc*, */etc/rcp3.d/K25rpc*, and */etc/rcp3.d/S75rpc*. Therefore, if powerfail occurs at run-level 2 or run-level 3, RPC is shut down and restarted automatically.

NOTE: Restarting RPC and NFS on the UNIX server may result in a new port number being assigned to the *pcnfsd* daemon. Some DOS clients communicating with the UNIX server via NFS may be unable to handle the new port assignment and have to be rebooted.

Customizing Memory Dumps

This section will describe some ways you can customize the notification of a dump in the dump area. To set up your system for dumps, see “Configuring the Dump Area” in the “Configuring Disks” chapter. To save the memory to the dump area or to save the dump area to media, refer to the appropriate procedures in Chapter 3, “Recovery Techniques”.

Changing the Default Response Time to Save Memory to Media

When a memory dump occurs, you have by default 60 seconds to save the system dump memory image before the system recovers from the memory dump. You can change this timeout value by modifying the value for `TIME` in the `/etc/default/dump` file. A value of `TIME=120` would cause the system to wait 120 seconds (2 minutes) before proceeding with system startup. A value of `TIME=0` would cause the system to proceed without prompting you to save the dump. A value of `TIME=-1` means the system waits indefinitely for you to save the dump (See the `dump(4)` and `dumpcheck(1M)` man pages).

Systems without Firmware Dump Capability

Enter the following command to test your system for firmware dump capability:

```
# machinetype -D
```

For systems that do not have firmware dump capability, you should set the value for `TIME` to `-1` in `/etc/default/dump`. This value allows you to save the dump without being restricted to a response time limit. After the dump has been saved, the system then tries to recover from the memory dump.

Systems with Firmware Dump Capability

For systems that have firmware dump capability, you can save the dump after the system has fully recovered from the dump. You do not have to choose to save the dump within the defined response time before the system recovers from the dump. All multiprocessor machines and those uniprocessor machines which have RBIOS have firmware dump capability.

Caution

If your dump and swap areas overlap, you should save the dump when prompted; that is, before the system performs a file system check of the root file system. In this case, you might want to set `TIME=-1` in the file `/etc/default/dump`. By doing this, the prompt does not time out when a system memory dump occurs. If you setup an expiring response time and dump and swap overlap, the dump is most likely lost after the timeout has expired. Systems with RBIOS type firmware whcih have dump set to the swap area will not timeout at the save prompt, regardless of the value for `TIME`.

Changing the Default Time Limit To Save the Dump Area to Media

If a memory dump occurs, you receive a message that there is a dump image in the dump device and that you should save it for analysis. You receive this message until you either save the dump, clear the dump flag, or the time limit to save the dump area to media expires.

This time limit to save the dump area to media can be found in `/etc/default/dump` and is called `FLAG_TIMEOUT` (see `dump(4)` and `dumpdetect(1M)`). The default value for `FLAG_TIMEOUT` is 24 hours. The maximum time for `FLAG_TIMEOUT` is 168 hours (7 days).

Common System Administration Tasks

It is important to save the dump or clear the dump flag as soon as possible. If another crash dump occurs or if the dump switch/button is activated before the dump flag is cleared, the following happens, depending on the version of firmware installed:

- For firmware versions prior to 1.03, the firmware overwrites the last dump.
- For firmware versions of 1.03 or greater, the firmware does NOT overwrite an existing dump that has not been saved. If you wish to save the new dump, interrupt the boot process and initiate a dump from the startup subsystem (SUS) menus (System Startup Services).

If a power outage occurs, the powerfail dump overwrites the crash memory image in the dump area(s) and clears the dump flag.

Also refer to “Saving Dump Area to Media” in the “Recovery Techniques” chapter.

Installing from a Remote Console (NCR 3450/3550 Only)

This section identifies the equipment requirements and procedures necessary to install the operating system software from a remote console.

Traditionally, the installation of the operating system software is performed through the local console, which is connected directly to a serial port on the system where the software is being installed.

Why/When

You can install the NCR UNIX SVR4 operating system software on an NCR Model 3450 or 3550 system from a console (Video Display Terminal) that is remotely connected to that system through modems and communication (telephone) lines.

The remote installation procedures may be performed from a remote support center or from a remote Help Desk, where the personnel are experienced with installation planning and procedures.

NOTE: Although the installation interaction (displays and responses) are performed by someone at the remote console, there must be someone present at the local system to set up and activate the remote console and to perform required media change operations.

Setting up the Equipment

Installing the operating system software from a remote console requires the following equipment to be set up correctly at both the local and remote sites.

Local Site Equipment Requirements

The following equipment must be properly installed at the local site (system being installed):

- NCR System 3000 Model 3450 or 3550
- VGA Console connected to the VGA monitor interface
- Modem connected to COM1 port

Remote Site Equipment Requirements

The following equipment must be properly installed at the remote site:

- Any VT100 emulating Video Display Terminal (VDT) or a VGA console on an NCR System 3000 computer
- Modem set up to match the local modem's baud rate, data bits and parity

Local Modem Requirements

The following specific local modem settings are required to permit remote installation of operating system software:

- Local echo **MUST** be disabled
- Result codes should be disabled
- Modem **MUST** be strapped to ignore DTR transitions
- Carrier should follow remote (Carrier Detect active only when local modem is connected to remote system/terminal)
- Auto Answer on
- 8 data bits/ no parity settings are recommended

Setting up the Remote Console

To set up the remote console, the local system operator must activate a system start up (reboot) and use the Start Up Subsystem (SUS) menus to perform the following procedure:

- Step 1:** Perform an orderly shutdown of the operating system software:

```
# shutdown -y -16 -g0
```

- Step 2:** To get to the SUS menus, you must interrupt the start up sequence when the following message appears:

Press spacebar to interrupt autoload sequence.

Press the spacebar within 5 seconds to go to the SUS menus. SUS then displays the **Startup Subsystem Services Main Menu**.

- Step 3:** From the **Startup Subsystem Services Main Menu**, select **Remote Support Services**.

- Step 4:** When SUS displays the **Remote Support Services** menu, select **Remote Console Setup**.

- Step 5:** SUS displays the **Remote Console Setup Configuration** menu to show the current console configuration information. This information includes: Console type, whether or not the remote console is active, baud rate, number of data bits, and parity.

This menu also displays the following question:

```
Do you wish to modify console device setup
(y/n): ?
```

- Step 6:** If the remote console characteristics are correct and if the remote console is currently **ACTIVE**, you may enter **n** and exit the console set up procedures.

If the remote console is **NOT ACTIVE** or if the characteristics are not correct, enter **y**.

Common System Administration Tasks

Step 7: If you answer **y** to the question in step 6, you must respond to the following question:

Remote Console operation is currently not active. Do you wish to activate it (y/n): ?

Step 8: Enter **y** to activate the remote console. SUS displays the current console type or "Not Installed." The display looks like the following:

Remote Console Type is currently "COM-1 RS-232 VDT"

1. COM-1 RS-232 VDT
2. Not Installed

Select a new console from the list?

Step 9: If the console is currently COM-1 RS-232 VDT, press **Return**; otherwise, select COM-1 RS-232 VDT, and then press **Return**. Next, SUS displays the baud rate selection.

Step 10: Select the baud rate from the list that SUS displays.

NOTE: 9600 baud modems are recommended, but if either modem is not 9600 baud, select the correct baud rate now.

When the current baud rate is correct, press **RETURN**. Next, SUS displays the number of bits per character selection.

Step 11: Select 8 bits per character, or press **RETURN** if 8 bits per character is the current setting. SUS displays the following:

Press spacebar to continue.

Step 12: Press the Spacebar. SUS displays the new **Remote Console Setup Configuration** menu and the following question:

Do you wish to adjust this console device
setup again (y/n) : ?

Step 13: If the displayed configuration is **NOT** correct, enter **y** and repeat the previous steps as required.

If the displayed configuration is correct, enter **n**, and press the Spacebar when requested to do so. SUS returns to the **Remote Console Setup** menu.

Step 14: Enter **<e>** to return to the **Startup Subsystem Services Main Menu**.

Step 15: The remote modem should call the local modem. After connection is established, the **Startup Subsystem Services Main Menu** and all following SUS menus and messages should appear on both the local and remote console terminals.

Step 16: The remote system operator notifies the local operator to insert Flex Diskette #1 (Volume 1).

Step 17: After the local system operator inserts Flex Diskette #1, either operator may select **System Boot Services** from the **Startup Subsystem Services Main Menu** displayed on the terminal.

Step 18: From the **System Boot Services** menu, select **System Processor Boot From Flex Disk**. The system bootloads the UNIX System from Flex Diskette #1, and then requests the operator to insert Flex Diskette #2.

Warning

After booting the UNIX System, although the local console displays the same messages as the remote console, the system will NOT accept input from the local console. The remote console is the ACTIVE console.

Step 19: After the local system operator inserts Flex Diskette #2, the remote system operator must select the type of installation to be performed: Interactive Installation or Installation With Defaults.

At this time, the remote system operator should continue at either the Interactive Installation or Installing With Defaults, and the local system operator should be available to make media changes as requested by the remote system operator.

Using the Maintenance and Reference Diskettes

This chapter describes the maintenance and reference diskettes as well as providing procedures for using them to maintain your system. It contains the following information:

What Are Maintenance and Recovery Diskettes?	2-2
Booting from a Flex Diskette	2-4
Copying Maintenance and Recovery Diskettes	2-6
Accessing the Maintenance File System	2-11

What Are Maintenance and Reference Diskettes?

You receive one set of diskettes with your computer hardware and another set with your operating system software.

The reference diskette is hardware-related and comes with the computer. You may receive an additional diskette if you add a piece of hardware, such as an Ethernet Controller.

Of the diskettes that come with your operating system tapes, three of them provide access to the maintenance file system. These are the two boot flex diskettes and the maintenance diskette.

Reference Diskette

The reference diskette is a 3.5-inch flex diskette that comes with your computer. This diskette contains utilities to configure the hardware, permit the system to work properly with various devices, copy the diskette, and perform a number of other tasks.

The *NCR 345x/35xx Unit Installation, Care and Cleaning* document that comes with your computer provides details about how to use this diskette. In addition, a number of procedures in this book use the Reference Disk. This chapter contains a procedure for copying the reference disk.

If you have the **madf** package installed on your system, you may use the Micro Channel configuration utilities instead of the reference diskette to configure Micro Channel adapters. See the “Configuring Adapters” chapter for instructions for using these utilities.

Boot Flex Diskettes

The two boot flex diskettes (Boot flexes 1 of 2 and 2 of 2) contain files that are essential to the proper installation and operation of your system. You **MUST** have them to install and to perform many of the necessary maintenance and recovery procedures. Some of the files necessary for installation and maintenance are located under the */install* file system on these diskettes.

Maintenance Diskette

The maintenance diskette (volume 3) contains additional utilities that you may want to use when performing maintenance operations. For example, the *vi* editor is on this diskette.

Performing Maintenance and Recovery

The “Recovery Techniques” chapter contains the following procedures that use the maintenance file system:

- Recovering files in */stand*
- Recovering *unix*
- Recovering the root password
- Recovering the */etc/passwd* and */etc/shadow* files
- Recovering files from the installation tape
- Recovering sector 0 on the root device
- Recovering the BOOT program
- Saving a memory dump

The “Restoring Information” chapter contains the following procedures that use the maintenance file system:

- Restoring an entire disk
- Restoring an entire system
- Restoring a corrupted root file system

Booting from a Flex Diskette

The procedures in this chapter require you to boot your system from flex diskettes (either the install flexes or the reference diskette). The procedure for booting from a flex diskette is different on uniprocessor and multiprocessor systems.

Uniprocessor Systems

Perform the following steps to boot a uniprocessor system from a flex diskette.

Step 1: Insert Boot flex 1 of 2 in the flex drive.

Step 2: Perform an orderly system shutdown:

```
.# shutdown -y -16 -gn
```

where n is the number of seconds until shutdown. When the shutdown completes, your system automatically reboots from the flex diskette.

Multiprocessor Systems

Perform the following steps to boot a multiprocessor system from a flex diskette.

Step 1: Insert Boot flex 1 of 2 in the flex drive.

Step 2: Perform an orderly system shutdown:

```
# shutdown -y -16 -gn
```

where n is the number of seconds until shutdown.

Using the Maintenance and Reference Diskettes

Step 3: When the following message appears, press the space bar to access the Startup Subsystem:

To interrupt autoload sequence, press the spacebar within 5 seconds . . .

The Startup Subsystem main menu appears on your screen.

Step 4: Select the following menu options:

System Boot Services
System Boot From Flex Disk

Your system then boots from the flex diskette.

Copying Maintenance and Reference Diskettes

You should always have at least one backup copy of the following diskettes.

Boot Flex Diskettes (Boot flex 1 of 2 and Boot flex 2 of 2)

These two diskettes contain the basic file system that is used for installation and maintenance. The diskettes are necessary for performing the recovery procedures.

Maintenance Diskette

This diskette contains additional utilities that you may want to use when performing maintenance operations.

Reference Diskette

This diskette is necessary for your system to properly identify the hardware and drivers. Whenever you make changes to the information on this diskette, you should make a new backup copy.

Caution

If one of these diskettes is lost or if the data on one of the diskettes is corrupted or accidentally overwritten, you can NOT perform many of the necessary maintenance and recovery procedures.

You should always use the latest working copy of the Reference Diskette to perform any of the maintenance and recovery tasks described in this chapter.

Whenever the contents of the working copy of the Reference Diskette change (for example, if you install ADF files for controller boards), you should make a new backup of the working copy.

Copying the Boot and Maintenance Flex Diskettes

To Copy the Diskettes:

Step 1: Insert the first boot flex diskette into the flex disk drive and copy the data from the flex diskette to a temporary file by entering the following command:

```
# dd if=/dev/rdisk/f03ht of=/tmp/file  
count=2880
```

Step 2: Remove the boot flex diskette from the flex disk drive.

Step 3: Insert a blank (writable) flex diskette that can be used as the working copy of the boot flex diskette.

Step 4: Format the flex diskette by entering the following command:

```
# format -i2 /dev/rdisk/f03ht
```

NOTE: Do NOT remove the working copy of the flex diskette.

Step 5: Copy the data from the temporary file to the flex diskette by entering the following command.

```
# dd if=/tmp/file of=/dev/rdisk/f03ht  
count=2880
```

Step 6: Make sure the flex diskette is write protected. Also, make sure you label the flex diskette as the working copy of the boot flex diskette.

Using the Maintenance and Reference Diskettes

Step 7: Repeat Steps 1 through 6 to copy the second boot flex diskette and the maintenance diskette.

To Test the Copies:

You should make sure that the working copies of the boot flex diskettes work properly by accessing the maintenance file system and mounting your *root* file system.

Step 1: Insert the copy of the first boot flex diskette (volume 1) into the flex disk drive.

Step 2: Reboot your system from flex diskette.

Step 3: When instructed, insert the copy of the second boot flex diskette (volume 2) into the flex disk drive and press the ENTER key.

Step 4: When instructed, confirm or change the date.

Step 5: When the following message appears, continue with installation by pressing 1.

Select one of the following:

1. Perform Installation
2. Perform System Maintenance
3. Perform System Restore
4. Perform Micro Channel Configuration

Type selection number, then press ENTER.>

Step 6: When the following message appears, press 3.

Select one of the following:

1. Interactive Installation
2. Installation with Default
3. Help
4. Cancel Installation

Type selection number, then press ENTER.>

Using the Maintenance and Reference Diskettes

Step 7: When the # prompt appears, type CTRL-d to return to the beginning of operations for the second flex. You are instructed to insert Boot Flex #2.

If you wish to continue with maintenance and mount the root file system, see “Accessing the Maintenance File System.”

If you do not want to continue maintenance, press DEL and CTRL-d.

Copying the Reference Diskette

- Step 1:** Insert the reference diskette in the flex disk drive.
- Step 2:** Reboot your system from the flex diskette.
- Step 3:** When the reference disk menu is displayed, select **Utilities**. Then select **Backup Disk**.
- Step 4:** Follow the instructions that appear on the screen; insert a blank and writable diskette in the flex disk drive, re-insert the reference diskette, etc. The copying process requires that you swap the flex diskettes approximately six (6) times.
- Step 5:** When the “Backup Complete” message appears, do NOT remove the working copy of the reference diskette.
- Step 6:** Press the ESC (escape) key as many times as necessary to exit the main menu. The system reboots automatically.
- Step 7:** Reboot the system from the flex diskette.
- Step 8:** If the working copy of the reference diskette works properly, remove the working copy from the flex disk drive and label the flex diskette as the working copy.

Using the Maintenance and Reference Diskettes

Step 9: Press the ESC key to exit the main menu. The system reboots automatically.

Accessing the Maintenance File System

You can use the maintenance file system to perform file system maintenance tasks. A third flex diskette contains additional utilities that you can use in performing maintenance and recovery tasks.

Examples of what you can do using the maintenance file system are to perform an `fsck` on the *root (/)* file system, recover *unix* and other files in */stand*, and recover the *BOOT* program.

Before You Start

- You should know the type of file system on the *root* device, either *s5* or *ufs*.
- If */usr* is a separate file system and you want to access the commands or utilities in */usr/bin* and */usr/sbin* from the maintenance file system, you must know the slice that contains */usr* and the file system type for */usr* so you can mount the */usr* file system to access the commands. This information is located in the */etc/vfstab* file in your root file system.

Procedure

To access the maintenance file system, use the following procedure.

Step 1: Insert the first boot flex diskette into the flex disk drive.

Using the Maintenance and Reference Diskettes

- Step 2:** Reboot the system from the flex diskette.
- Step 3:** When instructed, insert the second boot flex diskette (volume 2) into the flex disk drive and press the ENTER key.
- Step 4:** When the following message is displayed, access maintenance by pressing 2.

Select one of the following:

1. Perform Installation
2. Perform System Maintenance
3. Perform System Restore
4. Perform Micro Channel Configuration

Type selection number, then press ENTER.>

- Step 5:** When the following message appears, press Y followed by the ENTER key:

Would you like to mount the maintenance floppy disk?

- Step 6:** When instructed, insert the maintenance flex diskette (volume 3) into the flex disk drive and press the ENTER key.

NOTE: If the maintenance diskette is write-protected, the system can not mount it and generates an error.

NOTE: Do NOT remove the maintenance flex disk if you want to use the additional maintenance utilities.

You can make the flex drive available for copying to/from other diskettes by unmounting */maint* and removing the maintenance diskette. However, you must reinsert the maintenance diskette and mount it as */maint* before using the utilities again.

Using the Maintenance and Reference Diskettes

To unmount, enter:

```
# umount /maint
```

To remount, enter the following:

```
# mount -f /dev/dsk/f0 /maint
```

Step 7: When the shell prompt (#) appears, you can perform maintenance tasks on your file system.

NOTE: When */usr* is on a separate file system, remember to mount it as */mnt/usr*, if */mnt* is the root mount. To use utilities that exist on the mounted devices, remember to give the full path. For example if you wish to use *pg(1)* to look at the password file on the mounted root device, use the following command sequence:

```
# /mnt/usr/bin/pg /mnt/etc/passwd
```

NOTE: The tape device node is not included in the maintenance file system. If you need to access the tape device node, you must create it with the following command:

```
# mknod /dev/rmt/device_name c major minor
```

For example, if your tape device is *c0t3d0s0*, you would enter the following command:

```
# mknod /dev/rmt/c0t3d0s0 c 35 48
```

The default node has *yy* rewind options. If you require other rewind options, create those nodes as well.

NOTE: If you do not remember the major, minor numbers of your tape device, you may do either of the following:

- Mount your root slice and check under */dev/rmt*
- Enter the following command:

```
# nodes -d device_name
```

Exit the Maintenance File System

- Step 1: Exit the maintenance file system by pressing CTRL-d. A message asking if you are finished with system maintenance is displayed. The message also says that all mounted file systems will be unmounted automatically.
- Step 2: Answer y. The system automatically begins to reboot.
- Step 3: Remove the maintenance flex. (If you remove the flex before the system has unmounted it, you may corrupt its file system.)

NOTE: If your flex file system does get corrupted, enter the following command to fix it:

```
# fsck -F s5 /dev/dsk/f0
```

Recovering Files in the /stand File System	3-2
Recovering unix	3-6
Recovering the root Password	3-10
Recovering the /etc/passwd File	3-13
Recovering the /etc/shadow File	3-17
Recovering Files from the Installation Tape	3-21
Recovering Sector 0	3-25
Recovering the Boot Loader Program	3-29
Saving Memory to Dump Area	3-35
Saving Dump Area to Media	3-37
Examine the Contents of the Dump	3-39
Checking a File System from the Maintenance File System	3-40

NOTE: When you use command line procedures that include SCSI device names, be sure to substitute the appropriate device names for your particular system.

Recovering Files in the */stand* File System

There are five files in the *STAND* file system, which is mounted under the */stand* directory (and frequently called the */stand* file system).

Contents of */stand*

/stand contains the following files:

unix

ELF executable; a copy may exist in the */etc/conf/lf.d* directory on the *root* device.

boot

ASCII; a copy exists in the */etc/default* directory on the *root* device and in the maintenance file system.

at386

ELF executable; a copy exists in the */etc/initprog* directory on the *root* device and in the maintenance file system. If this file is missing, the following error messages are displayed on the console:

```
boot: Cannot load /etc/initprog/at386: file not
opened
boot: Cannot load initprog: /etc/initprog/at386
```

att

Not required; do not recover if missing or corrupted.

compaq

Not required; do not recover if missing or corrupted.

Procedure

Use the following procedure to recover the *boot* or *at386* files. If the *unix* file is missing or corrupted, you can recover it using the procedure described later in this chapter.

Access the Maintenance File System

- Step 1:** Insert the first boot flex diskette (volume 1) into the flex disk drive.
- Step 2:** Reboot the system from flex diskette.
- Step 3:** When instructed, insert the second boot flex diskette (volume 2) into the flex disk drive and press the ENTER key.
- Step 4:** When the following message is displayed, perform system maintenance by pressing **2**.

Select one of the following:

1. Continue Installation
2. Perform System Maintenance
3. Perform System Restore
4. Perform Micro Channel Configuration

Type selection number, then press ENTER.>

- Step 5:** When the following message is displayed, answer **n**.

Would you like to mount the maintenance
floppy disk (y)/n?

Recover Files

- Step 1:** When the shell prompt (#) appears, check the root file system.

There is no */dev/root* node in the maintenance file system. To determine the root file system, enter the following commands:

```
# cd /dev/dsk
# echo *
```

The device nodes that are present are the root file system nodes. Typically, the root file system is on slice 1.

Recovery Techniques

Enter one of the following commands, depending on the type of the *root* file system:

```
# /install/etc/fs/ufs/fsck /dev/dsk/device_name
```

– OR –

```
# /install/etc/fs/s5/fsck /dev/dsk/device_name
```

If you enter the wrong file system type, (for example, if */dev/dsk/device_name* is a ufs file system and you enter s5), an error message appears and the file system is NOT checked. Try again, entering the other file system type.

Step 2: Mount the *root* (/) file system using the */mnt* directory (mount point) as follows:

```
# mount -F ufs /dev/dsk/device_name /mnt
```

– OR –

```
# mount -F s5 /dev/dsk/device_name /mnt
```

Step 3: Check and, if necessary, repair the */stand* file system. For example, enter the following command:

```
# /install/etc/fs/bfs/fsck /dev/rdsk/device_name
```

Step 4: Create a new directory, such as */mnt1*, and then mount the */stand* file system. For example, enter the following commands:

```
# mkdir /mnt1
# mount -F bfs /dev/dsk/device_name /mnt1
```

See Appendix B, “Device Names and Numbers,” for an explanation of device names.

Step 5: Copy the file(s) from the *root* file system to the mounted bfs file system. For example:

```
# cp /mnt/etc/default/boot /mnt1/boot
```

```
# cp /etc/initprog/at386 /mnt1/at386
```

– OR –

```
# cp /mnt/etc/initprog/at386 /mnt1/at386
```

Exit the Maintenance File System

Step 1: Exit the maintenance file system by entering CTRL-d. A message asking if you are finished with system maintenance is displayed. The message also says that all mounted file systems will be unmounted automatically

Step 2: Answer y. The system will then automatically start rebooting.

Step 3: Remove the boot flex diskette from the drive.

Recovering *unix*

If the */stand/unix* or */stand/boot* file is missing, the following error messages appear on the console.

```
BFS not fully populated, REVERTING back to S5 1K
boot: Cannot open defaults file: /etc/default/boot
```

First, try to boot the system using the kernel */stand/unix.diag*, */stand/unix.old*, or */stand/unix.ncrm* and then rebuild or copy a kernel. To boot with an alternate kernel, press the space bar during system boot. A prompt appears; enter the name of the kernel you want to boot.

Procedure

If you can not boot the system, you can recover */stand/unix* as follows:

Access the Maintenance File System

- Step 1: Insert the first boot flex diskette (volume 1) into the flex disk drive.
- Step 2: Reboot the system from the flex diskette.
- Step 3: When instructed, insert the second boot flex diskette (volume 2) into the flex disk drive and press the ENTER key.
- Step 4: When the following message is displayed, perform system maintenance by pressing 2.

Select one of the following:

- 1. Continue Installation
- 2. Perform System Maintenance
- 3. Perform System Restore
- 4. Perform Micro Channel Configuration

Type selection number, then press ENTER.>

Step 5: When the following message is displayed, answer **n**.

```
Would you like to mount the maintenance
floppy disk (y)/n?
```

Recover Files

Step 1: When the shell prompt (**#**) appears, check the root file system.

There is no */dev/root* node in the maintenance file system. To determine the root file system, enter the following commands:

```
# cd /dev/dsk
# echo *
```

The device nodes that are present are the root file system nodes.

Enter one of the following commands, depending on the type of the *root* file system.

```
# /install/etc/fs/ufs/fsck /dev/dsk/device_name
```

– OR –

```
# /install/etc/fs/s5/fsck /dev/dsk/device_name
```

If you enter the wrong file system type, (for example, if */dev/dsk/device_name* is a ufs file system and you enter s5), an error message appears and the file system is NOT checked. Try again, entering the other file system type.

Step 2: Mount the *root* (/) file system using the */mnt* directory (mount point) as follows:

```
# mount -F ufs /dev/dsk/device_name /mnt
```

– OR –

```
# mount -F s5 /dev/dsk/device_name /mnt
```

See Appendix B, “Device Names and Numbers,” for an explanation of device names.

- Step 3: Make sure the *bfs* file system that contains the */stand/unix* file is not corrupted by performing a file system check. For example, enter:

```
# /install/etc/fs/bfs/fsck /dev/rdisk/device_name
```

See Appendix B, “Device Names and Numbers,” for an explanation of device names.

- Step 4: Create a new directory, such as */mnt1*, and then mount the */stand* file system. For example, enter the following commands:

```
# mkdir /mnt1
# mount -F bfs /dev/dsk/device_name /mnt1
```

See Appendix B, “Device Names and Numbers,” for an explanation of device names.

- Step 5: Change directories to the *root* file system.

```
# cd /mnt
```

- Step 6: Determine if there is a copy of *unix* in the *root* file system, such as in the */mnt/etc/conf/cf.d* directory.

- Step 7: Copy a valid *unix* into the */stand* file system by entering the appropriate commands:

- If there is a valid copy of *unix* in the *root* file system, copy that *unix*. You do not need to edit the */stand/boot* file. For example:

```
# cp /mnt/etc/conf/cf.d/unix /mnt1/unix
# sync
```

- If there is NOT a valid copy of *unix* in the *root* file system, use */etc/conf/cf.d/.unix.base*.

```
# cp /mnt/etc/conf/cf.d/.unix.base /mnt1/unix
# sync
```

- If the *.unix.base* file is NOT a valid copy of *unix*, copy *unix* from the maintenance file system:

```
# cp unix /mnt1/unix
# sync
```

NOTE: You must place the entry `rootdev=SCSI(x, 0)` in the */stand/boot* file to boot using the maintenance kernel. The value for *x* is the minor number of your root device.

Exit the Maintenance File System

Step 1: Exit the maintenance file system by entering CTRL-d. A message asking if you are finished with system maintenance is displayed. The message also says that all mounted file systems will be unmounted automatically.

Step 2: Answer *y*. The system will then automatically start rebooting.

Step 3: Remove the boot flex diskette from the drive.

Establish the New Kernel

Step 1: Log in as *root* and enter single user mode.

Step 2: Create a new kernel (current version) by entering the following command:

```
# /etc/conf/bin/ldbuild
```

Step 3: After the kernel is rebuilt, perform the following commands:

```
# sync
# shutdown -i6 -g0 -y
```

The new kernel is relinked and then the system is rebooted using the new kernel.

Recovering the root Password

If you forget the root password, you can use the maintenance file system on the boot flex diskettes to recover it.

Procedure

Access the Maintenance File System

- Step 1: Insert the first boot flex diskette (volume 1) into the flex disk drive.
- Step 2: Reboot the system from flex diskette.
- Step 3: When instructed, insert the second boot flex diskette (volume 2) into the flex disk drive and press the ENTER key.
- Step 4: When the following message is displayed, perform system maintenance by pressing 2.
- Select one of the following:
1. Continue Installation
 2. Perform System Maintenance
 3. Perform System Restore
 4. Perform Micro Channel Configuration
- Type selection number, then press ENTER.>
- Step 5: When the following message appears, press Y followed by the ENTER key:

Would you like to mount the maintenance floppy disk?

- Step 6: When instructed, insert the maintenance flex diskette (volume 3) into the flex disk drive and press the ENTER key.

Recover the Password

- Step 1: When the shell prompt (#) appears, check the *root* file system using **fsck**. Be sure to use the file system type and device name appropriate to your system.

There is no */dev/root* node in the maintenance file system. To determine the root file system, enter the following commands:

```
# ls -l /dev/rdisk/*
```

The device nodes that are present are the root file system nodes.

```
# fsck -F ufs /dev/rdisk/device_name
```

– OR –

```
# fsck -F s5 /dev/rdisk/device_name
```

If you enter the wrong file system type, (for example, if *root* is a ufs file system and you enter s5), an error message appears and the file system is NOT checked. Try again, entering the other file system type.

- Step 2: Mount the *root* (/) file system using the */mnt* directory (mount point). For example, enter:

```
# mount -F ufs /dev/dsk/device_name /mnt
```

– OR –

```
# mount -F s5 /dev/dsk/device_name /mnt
```

- Step 3: Edit the */mnt/etc/shadow* file to remove the encrypted password for the *root* login.

Step 4: Check the */mnt/etc/passwd* file to be sure the password field for the **root** login is blank or contains the letter **x**.

- If the password field for the **root** login in the */etc/passwd* file contains the letter **x** (as it usually does), you are not prompted for a password when you log in as **root**. After you log in, use the **passwd** command to set a new password for **root**.
- If the password field for the **root** login in the */etc/passwd* file is blank, you are not prompted to enter a password.

Exit the Maintenance File System

Step 1: Exit the maintenance file system by entering **CTRL–d**. A message asking if you are finished with system maintenance is displayed. The message also says that all mounted file systems will be unmounted automatically.

Step 2: Answer **y**. The system will then automatically start rebooting.

Step 3: Remove the maintenance flex diskette from the drive.

Recovering the */etc/passwd* File

If you can not log in because your */etc/passwd* file is missing or corrupted, you can recover the */etc/passwd* file using the install boot flexes.

Before You Start

Read the “Maintaining the Password and Shadow Files” in Chapter 1 if you are not already familiar with the *passwd* and *shadow* files.

Procedure

Access the Entire Maintenance File System

- Step 1: Insert the first boot flex diskette (volume 1) into the flex disk drive.
- Step 2: Reboot the system from flex diskette.
- Step 3: When instructed, insert the second boot flex diskette (volume 2) into the flex disk drive and press the ENTER key.
- Step 4: When the following message is displayed, perform system maintenance by pressing 2.

Select one of the following:

- 1. Continue Installation
- 2. Perform System Maintenance
- 3. Perform System Restore
- 4. Perform Micro Channel Configuration

Type selection number, then press ENTER.>

- Step 5: When the following message appears, press **Y** followed by the **ENTER** key:

Would you like to mount the maintenance floppy disk?

- Step 6: When instructed, insert the maintenance flex diskette (volume 3) into the flex disk drive and press the **ENTER** key.

Recover /etc/passwd

- Step 1: When the shell prompt (**#**) appears, check the *root* file system using **fsck**. Be sure to use the file system type and device name appropriate to your system.

There is no */dev/root* node in the maintenance file system. To determine the root file system, enter the following command:

```
# ls -l /dev/rdisk/*
```

The device nodes that are present are the root file system nodes.

```
# fsck -F ufs /dev/rdisk/device_name
```

– OR –

```
# fsck -F s5 /dev/rdisk/device_name
```

If you enter the wrong file system type, (for example, if *root* is a *ufs* file system and you enter *s5*), an error message appears and the file system is **NOT** checked. Try again, entering the other file system type.

- Step 2: Mount the *root* (/) file system using the */mnt* directory (mount point). For example, enter:

```
# mount -F ufs /dev/dsk/device_name /mnt
```

– OR –

```
# mount -F s5 /dev/dsk/device_name /mnt
```

- Step 3:** Save your current password file if it exists. You may need to obtain information from it when you exit the maintenance file system.

```
# cd /mnt
# cp /mnt/etc/passwd /mnt/etc/savepasswd
```

- Step 4:** Determine if the */mnt/etc/opasswd* file exists.

- If it does, copy it to */mnt/etc/passwd*.

```
# cp /mnt/etc/opasswd /mnt/etc/passwd
```

- If it does not, then copy the */etc/passwd* file from the maintenance file system into the *root* file system.

```
# cp /etc/passwd /mnt/etc/passwd
```

- Step 5:** Check the password field for the *root* login in the */mnt/etc/passwd* file.

The password field is the one following the login name. For example if you login as *root*, the password field should look similar to the following:

```
root:x:0:1:0000-Admin(0000):/:
```

- Step 6:** Edit the field so that it contains either the letter *x* or is blank.

- Step 7:** Check the password field for the *root* login in the */mnt/etc/shadow* file.

Again, the password field is the one following the login name. For example:

```
root::7567:0:168:7:::
```

- Step 8:** Edit the password field so that it is blank.

Exit the Maintenance File System

- Step 1:** Exit the maintenance file system by entering CTRL-d. A message asking if you are finished with system maintenance is displayed. The message also says that all mounted file systems will be unmounted automatically.
- Step 2:** Answer y. The system will then automatically start rebooting.
- Step 3:** Remove the maintenance flex diskette from the drive.
- If you copied an *opasswd* file, any user trying to access the system with a newer password is unable to log in. If this happens, remove the encrypted password from the */etc/shadow* file so the user can set a new password.
 - If you copied the *passwd* file from the */install* file system, you must now recover an */etc/passwd* file from a recent backup or edit the file you copied to add additional logins such as those for users. Be sure you maintain the original user and group IDs for the users.

Recovering the */etc/shadow* File

If you can not log in because your */etc/shadow* file is missing or corrupted, you can recover the */etc/shadow* file using the maintenance file system.

Before You Start

Read “Maintaining the Password and Shadow Files” in Chapter 1 if you are not already familiar with the *passwd* and *shadow* files.

Access the Entire Maintenance File System

- Step 1: Insert the first boot flex diskette (volume 1) into the flex disk drive.
- Step 2: Reboot the system from flex diskette.
- Step 3: When instructed, insert the second boot flex diskette (volume 2) into the flex disk drive and press the ENTER key.
- Step 4: When the following message is displayed, perform system maintenance by pressing 2.

Select one of the following:

- 1. Continue Installation
- 2. Perform System Maintenance
- 3. Perform System Restore
- 4. Perform Micro Channel Configuration

Type selection number, then press ENTER.>

NOTE: Do not remove your second boot flex.

Step 5: When the following message appears, press **Y** followed by the **ENTER** key:

Would you like to mount the maintenance floppy disk?

Step 6: When instructed, insert the maintenance flex diskette (volume 3) into the flex disk drive and press the **ENTER** key.

Recover /etc/shadow

Step 1: When the shell prompt (**#**) appears, check the *root* file system using **fsck**. Be sure to use the file system type and device name appropriate to your system.

There is no */dev/rroot* node in the maintenance file system. To determine the root file system, enter the following command:

```
# ls -l /dev/rdisk/*
```

The device nodes that are present are the root file system nodes.

```
# fsck -F ufs /dev/rdisk/device_name
```

– OR –

```
# fsck -F s5 /dev/rdisk/device_name
```

Step 2: Mount the *root (/)* file system using the */mnt* directory (mount point). For example, enter:

```
# mount -F ufs /dev/dsk/device_name /mnt
```

– OR –

```
# mount -F s5 /dev/dsk/device_name /mnt
```

Step 3: Save your current password shadow file, if it exists. You may need to obtain information from it later.

```
# cd /mnt
# cp /mnt/etc/shadow /mnt/etc/saveshadow
```

Step 4: Use a system editor to create an */mnt/etc/shadow* file that contains the following entry for the *root* login:

```
root:::::::::
```

NOTE: Exactly 8 colons must follow the word “root.”

Step 5: Check the */mnt/etc/passwd* file to be sure the password field for the *root* login is blank or contains the letter x.

Exit the Maintenance File System

Use the following procedure to exit the maintenance file system:

Step 1: Exit the maintenance file system by entering CTRL-d. A message asking if you are finished with system maintenance is displayed. The message also says that all mounted file systems will be unmounted automatically.

Step 2: Answer y. The system will then automatically start rebooting.

Step 3: Remove the maintenance flex diskette from the drive.

Step 4: You can now log in as **root**. For your system to be fully accessible, you need to perform some additional recovery.

- If you have an */etc/oshadow* file, copy that file to */etc/shadow*. If you have changed the **root** password since */etc/oshadow* was created, remove the encrypted password for the **root** login so you can set a new one when you log in again.
- If you do not have an */etc/oshadow* file, you must either recover an */etc/shadow* file from a recent backup (or from information in the saved corrupted file) or you must run the **pwconv(1M)** command to create the necessary additional entries in the existing */etc/shadow* file.

The **pwconv(1M)** command creates entries with a locked password (the letter **x** in the password field). You must remove the letter **x**, leaving a blank field, before anyone can use the login.

Recovering Files from the Installation Tape

You can use the following procedures to recover one or more files from the installation (base files) tape. Which procedure you use depends on whether your operating system is running.

If Your Operating System Is Running

- Step 1: Log in as **root** and access the operating system shell.
- Step 2: Insert the installation (base files) tape into the cartridge tape drive.
- Step 3: Move to the *root* directory of the *root* file system by performing the following command:
- ```
cd /
```
- Step 4: Transfer the package containing the operating system base files (BASE) into a temporary directory (for example, */tmp*).
- ```
# mkdir /tmp/base
# pkgtrans -d /dev/rmt/device_name /tmp/base \
BASE
```
- See Appendix B, “Device Names and Numbers,” for an explanation of device names.
- Step 5: Copy the desired file(s) from the temporary location to the proper directory.
- ```
cp /tmp/base/etc/conf/bin/ldbuild \
/etc/conf/bin/ldbuild
```



Step 6: When you have copied the files you want, remove the base files from the temporary directory. For example:

```
rm -r /tmp/base
```

Step 7: Use the `ls -l` command to check the permissions, size, etc., of the recovered files.

---

### If Your Operating System Is Not Running

#### Access the Maintenance File System

Step 1: Insert the first boot flex diskette (volume 1) into the flex disk drive.

Step 2: Reboot the system from flex diskette.

Step 3: When instructed, insert the second boot flex diskette (volume 2) into the flex disk drive and press the ENTER key.

Step 4: When the following message is displayed, perform system maintenance by pressing 2.

```
Select one of the following:
```

1. Continue Installation
2. Perform System Maintenance
3. Perform System Restore
4. Perform Micro Channel Configuration

```
Type selection number, then press ENTER.>
```

Step 5: When the following message appears, press y, followed by the ENTER key.

```
Would you like to mount the maintenance
floppy disk?
```

Step 6: When instructed, insert the maintenance flex diskette (volume 3) into the flex drive and press the ENTER key.

## Restore Files

**Step 1:** When the shell prompt (#) appears, enter one of the following commands, depending on the type of the *root* file system.

There is no */dev/rroot* node in the maintenance file system. To determine the root file system, enter the following command:

```
ls -l /dev/rdisk/*
```

The device nodes that are present are the root file system nodes.

```
/install/etc/fs/ufs/fck /dev/rdisk
/device_name
```

– OR –

```
/install/etc/fs/s5/fck /dev/rdisk/device_name
```

If you enter the wrong file system type, (for example, if */dev/rroot* is a ufs file system and you enter s5), an error message appears and the file system is NOT checked. Try again, entering the other file system type.

**Step 2:** Mount the *root* (/) file system using the */mnt* directory (mount point) as follows:

```
mount -F ufs /dev/dsk/device_name /mnt
```

– OR –

```
mount -F s5 /dev/dsk/device_name /mnt
```

**Step 3:** Insert the installation (base files) tape into the cartridge tape drive.

**Step 4:** Move to the *root* directory of the *root* file system by entering the following command:

```
cd /mnt
```

- Step 5: Transfer the package containing the operating system base files (BASE) into a temporary directory (for example, */tmp*).

```
mkdir /mnt/tmp/base
/mnt/usr/bin/pkgtrans -d \
/dev/rmt/device_name /mnt/tmp/base BASE
```

See Appendix B, “Device Names and Numbers,” for an explanation of device names.

- Step 6: Copy the desired file(s) from the temporary location to the proper directory.

```
cp /mnt/tmp/base/etc/conf/bin/ldbuild \
/mnt/etc/conf/bin/ldbuild
```

### Exit the Maintenance File System

- Step 1: Exit the maintenance file system by entering CTRL-d. A message asking if you are finished with system maintenance is displayed. The message also says that all mounted file systems will be unmounted automatically.
- Step 2: Answer y. The system will then automatically start rebooting.
- Step 3: Remove the maintenance flex diskette from the drive.

## Recovering Sector 0

Sector 0 on the *root* device contains the disk configuration information which can be displayed and/or recovered using the **fdisk** utility.

---

### Contents of Sector 0

Sector 0 contains the following information:

#### *Partition*

The number of the partition on the disk that contains all files needed to boot the system. The default value is 1 which indicates that partition 1 (*ROOT* file system) contains the files needed to boot the system.

#### *Status*

Indicates the status of the partition, such as *Active*.

#### *Type*

Indicates the type of partition, such as *UNIX Sys*.

#### *Start*

Indicates the starting sector for that partition.

#### *End*

Indicates the ending sector for that partition.

#### *Length*

Indicates the length (number of sectors) for the partition.

#### *%*

Indicates the percentage of the disk used for the partition, such as 100%.

---

### Using the fdisk Utility

You can use the **fdisk** utility to change the data in sector 0; the **fdisk** utility is available in both the maintenance file system and the *root* file system (*/usr/bin* directory).

When you perform the **fdisk** utility, the current disk configuration is displayed. Then, the following menu appears:

SELECT ONE OF THE FOLLOWING:

- 1.Create a partition
  - 2.Change Active (Boot from) partition
  - 3.Delete a partition
  - 4.Exit (Update disk configuration and exit)
  - 5.Cancel (Exit without updating disk configuration)
- Enter Selection:

If sector 0 is missing or corrupted, the following error message appears on the console:

```
No Boot Sector on Fixed Disk - ROM BASIC is not
supported.
Press <F1>
```

If you press function key 1 (F1), the system performs a retry and the same error message appears.

---

### Procedure

To recover sector 0, perform the following procedure:

#### Access the Entire Maintenance File System

- Step 1: Insert the first boot flex diskette (volume 1) into the flex disk drive.
- Step 2: Reboot the system from flex diskette.
- Step 3: When instructed, insert the second boot flex diskette (volume 2) into the flex disk drive and press the ENTER key.

**Step 4:** When the following message is displayed, perform system maintenance by pressing 2.

```
Select one of the following:
 1. Continue Installation
 2. Perform System Maintenance
 3. Perform System Restore
 4. Perform Micro Channel Configuration
Type selection number, then press ENTER.>
```

**Step 5:** When the following message appears, press y, followed by the ENTER key.

```
Would you like to mount the maintenance
floppy disk?
```

**Step 6:** When instructed, insert the maintenance flex diskette (volume 3) into the flex drive and press the ENTER key.

### Recover Sector 0

**Step 1:** Enter the `fdisk` command to partition the disk. For example:

```
fdisk /dev/rdisk/device_name
```

See Appendix B, “Device Names and Numbers,” for an explanation of device names.

**Step 2:** The recommended configuration (default) is 100% UNIX. Therefore, enter the letter y (yes) to configure the disk with the default values.

**Step 3:** Enter the `fdisk` command again to check the disk configuration. If necessary, change the default disk configuration to match your configuration before the data was corrupted or missing.

### Exit the Maintenance File System

**Step 1:** Exit the maintenance file system by entering CTRL-d. A message asking if you are finished with system maintenance is displayed. The message also says that all mounted file systems will be unmounted automatically.

## Recovery Techniques

Step 2: Answer **y**. The system will then automatically start rebooting.

Step 3: Remove the maintenance flex diskette from the drive.

### Caution

If you create a DOS partition, the **prtvtoc** utility displays that partition as *NONUNIX* in slice 5. If you log in as **root**, you can create a file system in that partition; however, you destroy the data in that DOS partition.

## Recovering the Boot Loader Program

The boot loader consists of three parts. The first two parts of the boot loader program are found in the */etc/boot* file, and the third part is found in the */stand/ThirdStage* file. The information in the */etc/boot* file is also written on slice 7 on the *root* device, in blocks 1–28. When the system starts up, and the master boot block (block 0) has determined the active partition is UNIX, then the boot program contained in blocks 1–28 is responsible to read the disk slicing information contained in block 29 of slice 7 and determine the location of the */stand* slice. The file */stand/ThirdStage* is then called by the boot program and loads the kernel */stand/unix* into memory. The two files */etc/boot* and */stand/ThirdStage* must be from the same operating system release. (Block 30–34 of slice 7 are currently unused).

If the boot program is missing or corrupted, the following error message appears on the console when you try to boot the system:

```
Missing operating system
```

Also, if the boot program and *ThirdStage* do not match, the system hangs with the following message on the screen:

```
Booting UNIX . . .
```

You can use the following procedure to restore the boot program.



---

## Procedure

### Access the Entire Maintenance File System

**Step 1:** Insert the first boot flex diskette (volume 1) into the flex disk drive.

**Step 2:** Reboot the system from flex diskette.

**Step 3:** When instructed, insert the second boot flex diskette (volume 2) into the flex disk drive and press the ENTER key.

**Step 4:** When the following message is displayed, perform system maintenance by pressing 2.

Select one of the following:

1. Continue Installation
2. Perform System Maintenance
3. Perform System Restore
4. Perform Micro Channel Configuration

Type selection number, then press ENTER.>

**Step 5:** When the following message appears, press y, followed by the ENTER key.

Would you like to mount the maintenance floppy disk?

**Step 6:** When instructed, insert the maintenance flex diskette (volume 3) into the flex drive and press the ENTER key.

## Recover the BOOT Program

The following device names will be used in this procedure:

- *device\_name* = whole disk (same as slice 0)
- *device\_name1* = root slice
- *device\_name2* = stand slice

Step 1: When the shell prompt (#) appears, check the root file system.

There is no */dev/root* node in the maintenance file system. To determine the root file system, enter the following command:

```
ls -l /dev/rdisk/*
```

The device nodes that are present are the root file system nodes.

Enter one of the following commands, depending on the type of the *root* file system.

```
/install/etc/fs/ufs/fsck /dev/rdisk/device_name1
```

– OR –

```
/install/etc/fs/s5/fsck /dev/rdisk/device_name1
```

If you enter the wrong file system type, (for example, if */dev/root* is a ufs file system and you enter s5), an error message appears and the file system is NOT checked. Try again, entering the other file system type.

Step 2: Mount the *root (/)* file system using the */mnt* directory (mount point) as follows:

```
mount -F ufs /dev/dsk/device_name1 /mnt
```

– OR –

```
mount -F s5 /dev/dsk/device_name1 /mnt
```

Step 3: To recover the BOOT program, first check the existence of the BOOT file in the root file system:

```
ls /mnt/etc/boot
```

If this file exists, you can recover the BOOT program from the backup copy on your disk. Copy the boot program */mnt/etc/boot* from the root file system on the hard disk to blocks 1–28 of the BOOT slice 7 on the disk:

```
dklayout -d /dev/rdisk/device_name \
-b -f /mnt/etc/boot
```

See Appendix B, “Device Names and Numbers,” for an explanation of device names.

If this file has been deleted or is corrupt, and you have a current backup of your root file system you must restore the BOOT file from your last backup before you can recover the BOOT program. Your backup must contain the same operating system version as your current system. If your system has any operating system upgrade software or operating system patches installed after the initial installation, the backup must also include any of those. Use following commands:

```
cd /mnt
cpio -ivd </dev/rmt/backup_device_name "etc/boot"
```

Copy the recovered boot program from */mnt/etc/boot* in the root file system on the hard disk to blocks 1–28 of the BOOT slice 7 on the disk:

```
dklayout -d /dev/rdisk/device_name \
-b -f /mnt/etc/boot
```

**NOTE:** The *device\_name* must specify slice 0 of a disk device.

If this file has been deleted or is corrupt, and you do NOT have a current backup of your root file system, you must recover the BOOT program from the maintenance file system. The following command copies the BOOT program from the maintenance file system to the blocks 1–28 in BOOT slice 7:

```
dklayout -d /dev/rdisk/device_name \
-b -f /install/etc/boot
```

Copy the BOOT program also to the root file system on disk into the */etc* directory, so there is always a backup copy available again on the system:

```
cp /install/etc/boot /mnt/etc/boot
```

- Step 4: Create a directory (i.e. *stand*) under the */mnt* directory and mount the */stand* slice from your root disk to that directory. Use the following commands:

```
mkdir /mnt/stand
mount -F bfs /dev/dsk/device_name2 /mnt/stand
```

- Step 5: To have a complete boot loader program, copy the *ThirdStage* file from maintenance floppy to your */stand* file system on the root disk. Use the following command:

```
cp /install/tmp/ThirdStage
/mnt/stand/ThirdStage
```

Step 6: Unmount the */stand* file system, using these commands:

```
cd /
umount /mnt/stand
```

### Exit the Maintenance File System

- Step 1: Exit the maintenance file system by entering CTRL-d. A message asking if you are finished with system maintenance is displayed. The message also says that all mounted file systems will be unmounted automatically.
- Step 2: Answer **y**. The system will then automatically start rebooting.
- Step 3: Remove the maintenance flex diskette from the drive.

## Saving Memory to Dump Area

Note that the dump area can be spread across multiple slices and/or disk drives. The total size of the dump area must be at least equal to the size of physical memory.

---

### System Panics

If your system panics and `ctkdb` is not installed, memory is automatically saved to the dump area if the dump flag is not set. If you are aware of when the system panicked, record the date and time you suspect the system stopped.

If `ctkdb` is installed, a `ctkdb` prompt (`K0>`) appears. Use the following procedure to obtain a memory dump:

- Step 1: Record the date and time you suspect the system stopped.
- Step 2: Record any information on the screen.
- Step 3: Type `go` at the `ctkdb` prompt. The system writes the contents of memory to the dump area.
- Step 4: Perform the steps described in the procedure “Saving Dump Area to Media”.

---

### System Hangs

If you suspect that your system is hanging, you must initiate the memory dump. To do this, follow these procedures:

#### On Machines with Dump Switch/Button:

- Step 1: Record the date and time you suspect the system stopped.

- Step 2: Record any information on the console screen.
- Step 3: Activate the dump switch/button on your system. On the NCR 3450, press the reset button located on the front of the unit. On the NCR 3550, toggle the dump switch located at the bottom of the Local Peripheral Board (LPB).

If **ctkdb** is not installed on your system, the system automatically tries to perform the memory dump.

If **ctkdb** is installed on your system, activating the dump switch/button results in a **ctkdb** prompt. If a **ctkdb** prompt appears, activate the dump switch/button again to cause a warm start followed by a memory dump.

- Step 4: Perform the steps described in the procedure “Saving Dump Area to Media”.

### **On Machines Without Dump Switch/Button:**

- Step 1: Record the date and time you suspect the system stopped.
- Step 2: Record any information on the console screen.
- Step 3: If **ctkdb** is installed, activate **ctkdb** by pressing the CTRL-ALT-D keys. Then type the following command at the **ctkdb** prompt:

**K0> sysdump**

If **ctkdb** is not installed, press the SHIFT-CTRL-ALT keys and type **dump**. You must actually type the word **dump** while pressing the SHIFT, CTRL and ALT keys.

- Step 4: Perform the steps described in the procedure “Saving Dump Area to Media”.

## Saving Dump Area to Media

---

### After the System Recovers from Memory Dump

The system notifies you automatically if there is a system memory dump which has not been saved. If a memory dump occurs, the following message appears:

**WARNING: THERE IS A SYSTEM MEMORY DUMP IMAGE IN THE DUMP DEVICE. PLEASE SAVE THE DUMP FOR ANALYSIS.**

This message continues to appear for a configurable time limit until one of the following situations becomes true:

- You have saved the dump
- The time limit as specified by the variable `FLAG_TIMEOUT` as defined in the file `/etc/default/dump` has expired (See “Customizing Memory Dumps” in Chapter 1, “Being a System Administrator”).
- The dump flag is cleared

If you do not save the dump nor clear the dump flag and the time limit expires, an error is logged and the following message appears:

**ALERT: DUMP FLAG CLEARED. ADMINISTRATOR NEGLECTED TO SAVE DUMP.**



**NOTE:** On systems with Firmware Release 1.03 or later, any subsequent crash memory dumps are lost until the dump flag is cleared. The dump flag is automatically cleared when you save the dump, when the `FLAG_TIMEOUT` expires, or a powerfail dump occurs. Also, you can choose to manually clear the dump flag using the command **`dumpsave -c`**.

---

### Saving the Dump Area to Media BEFORE Rebooting

When the system is rebooted following a system crash or manual dump, you are informed that a dump image is in the dump area. You must respond the message within 60 seconds (default value for `TIME` in the file `/etc/default/dump`; refer to “Customizing Memory Dumps” in Chapter 1, “Being a System Administrator”) or it times out and the startup continues without saving the memory dump. You may select the media on which you wish to save the dump.

When the dump is saved to media, the operating system automatically writes out the current kernel to the media. It asks you if `/stand/unix` is the UNIX running at the time of the dump; if not, the system prompts you to enter the name of the running kernel.

**NOTE:** If your dump area is also used as the swap device, you must save the dump area BEFORE the system is rebooted or the dump image is overwritten.

---

### Saving the Dump Area to Media AFTER Rebooting

If you did not save the dump during the reboot of the system you may login to the system and use the **`dumpsave`** utility to create the dump media. See **`dumpsave(1M)`** for complete details on usage and diagnostics.

The **`dumpsave`** utility automatically writes out the current kernel to tape. It asks you if `/stand/unix` is the UNIX running at the time of the dump; if not, the system prompts you to enter the name of the running kernel.

## Examine the Contents of the Dump

After the dump image has been saved using **dumpsave**, the dump and kernel may then be loaded using **ldsysdump** command. For details refer to the Reference Manual for **ldsysdump(1M)**. Have a systems analyst or support personnel analyze the dump to find out the reason for the crash.

## Checking a File System from the Maintenance File System

If a system panic has badly corrupted your system file systems (*/*, */stand*, */usr*, */var*) so you can not boot your system, you may need to check it with **fsck** from the maintenance file system.

---

### Access the Entire Maintenance File System

**Step 1:** Insert the first boot flex diskette (volume 1) into the flex disk drive.

**Step 2:** Reboot the system from flex diskette.

**Step 3:** When instructed, insert the second boot flex diskette (volume 2) into the flex disk drive and press the ENTER key.

**Step 4:** When the following message is displayed, perform system maintenance by pressing 2.

Select one of the following:

1. Continue Installation
2. Perform System Maintenance
3. Perform System Restore
4. Perform Micro Channel Configuration

Type selection number, then press ENTER.>

**Step 5:** When the following message appears, press Y followed by the ENTER key:

Would you like to mount the maintenance floppy disk?

- Step 6: When instructed, insert the maintenance flex diskette (volume 3) into the flex disk drive and press the ENTER key.

## Check the Root File System

- Step 1: When the shell prompt (#) appears, check the *root* file system using **fsck**. Be sure to use the file system type and device name appropriate to your system.

There is no */dev/root* node in the maintenance file system. To determine the root file system, enter the following command:

```
ls -l /dev/rdisk/*
```

The device nodes that are present are the root file system nodes.

```
fsck -F ufs /dev/rdisk/device_name
```

– OR –

```
fsck -F s5 /dev/rdisk/device_name
```

If you enter the wrong file system type, (for example, if *root* is a ufs file system and you enter s5), an error message appears and the file system is NOT checked. Try again, entering the other file system type.

## Exit the Maintenance File System

Use the following procedure to exit the maintenance file system:

- Step 1: Exit the maintenance file system by entering CTRL-d. A message asking if you are finished with system maintenance is displayed. The message also says that all mounted file systems will be unmounted automatically

## Recovery Techniques

**Step 2:** Answer **y**. The system will then automatically start rebooting.

**Step 3:** Remove the maintenance flex diskette from the drive.

This chapter describes general file system administration commands and the setup of standard disk file systems. Once the file systems are installed and being used you should read the “Disk Utilization” chapter in Volume 3 to find out about maintenance tasks on file systems. The tasks in this chapter include the following:

|                                              |      |
|----------------------------------------------|------|
| General Administration of File Systems ..... | 4-2  |
| Making (Creating) File Systems .....         | 4-7  |
| Mounting File Systems .....                  | 4-11 |
| Unmounting File Systems .....                | 4-13 |
| Displaying File System Information .....     | 4-15 |
| Compacting Files in /stand .....             | 4-18 |
| Using Quotas .....                           | 4-20 |

This chapter does not discuss the */proc* file system (refer to **proc(4)** in the *NCR UNIX SVR4 MP-RAS Reference Manual*). Appendix A describes the file system hierarchy and the possible file system types. It also describes the *ufs*, *s5*, and *bfs* type file systems in detail.

## General Administration of File Systems

This section describes generic commands, files, and procedures that apply to all (or many) file system types.

---

### Maintaining Different Types of File Systems

The Virtual File System architecture allows multiple file system types (FSTypes) to coexist in the kernel. Each FSType has specific features that are not shared with any other FSTypes; however, the file system administrative commands provide a common interface that allows you to maintain different types of file systems.

You can use the following commands to administer the FSTypes:

- **dcopy(1M)**
- **df(1M)**
- **ff(1M)**
- **fsck(1M)**
- **fsdb(1M)**
- **fstyp(1M)**
- **labelit(1M)**
- **mkfs(1M)**

- **mount(1M)**
- **mountall(1M)**
- **ncheck(1M)**
- **umount(1M)**
- **umountall(1M)**

Most of these commands can be invoked as follows (note that the line is split for readability):

```
command [-F FSType] [-V] [current_options] \
[-o specific_options] operands
```

**-F**

Specifies the *FSType* on which the command acts. If omitted, the *FSType* must be determinable from */etc/vfstab* (see “Using the *vfstab* File” in this chapter).

**-V**

Causes the command to echo the command line. The echoed line includes information derived from the *vfstab* file. You can use the **-V** option to verify the command line; it does not cause the command to execute.

*current\_options*

Options supported by the *s5*-specific module of the command.

**-o**

Specifies *FSType*-specific options. The *specific\_options* are a comma-separated list of keywords and/or keyword-attribute pairs to be interpreted by the *FSType*-specific module of the command.

*operands*

*FSType*-specific information. Refer to the appropriate *FSType*-specific page in the *Reference Manual* for the command.



**NOTE:** If you want to look up the FSType-specific information about a general file system administration command in the on-line manual, you must add an underscore to the command and the file system type for which you want specific information. Examples include:

```
man mkfs_s5
man mount_cdfs
man fsck_ufs
man fsdb_ufs
man fsck_bfs
```

---

### Using the *vfstab* File

The */etc/vfstab* file is an ASCII file that contains a list of default parameters for each file system.

Since the generic commands work on multiple FSTypes, they require FSType-specific information that can be provided on the command line or through the file system table */etc/vfstab*.

Each line in the file contains information about a file system in the following format:

```
special fsckdev mountp fstype fsckpass automnt mntopts
```

The meaning of each field is:

*special*

Block special device (for local devices).

*fsckdev*

Character special device that corresponds to *special*.

*mountp*

Default mount directory (mount point).

*fstype*

Type of file system (FSType).

*fsckpass*

Pass number used by **ff**, **fsck**, and **ncheck** to decide whether to check the file system automatically. Use **-** to inhibit automatic checking.

*automnt*

Whether the file system should be mounted automatically when the system is booted (yes or no).

*mntopts*

List of comma-separated options used in mounting the file system (see **mount(1M)** in the *Reference Manual* for available options). Use **-** to indicate no options.

**NOTE:** If the **support** package is installed on your system and you specified to use manual **fsck** during installation time of the package, any flag *fsckpass* in the file */etc/vfstab* set to automatic file system checking at startup time is overridden. If you have the **support** package installed and if you specified to use automatic file system checking at startup time, the only purpose of the *fsckpass* is to be able exclude specific file systems from an automatic file system check.

**NOTE:** However, if the **support** package is not installed on your system, the *fsckpass* flag will specify whether or not to perform an automatic file system check.

---

### Listing Installed File System Types

You can use the **crash(1M)** utility to display a list of FSTypes installed in the kernel. In addition to the familiar FSTypes such as **s5**, **crash** also lists internal FSTypes such as **specfs**.

To display a list of FSTypes installed in the kernel, enter the following command sequence:

```
crash <<!
> vfssw
> !
```

---

### Identifying the Type of a File System

Most commands used to administer files require you to specify the FSType on the command line or have it in the */etc/vfstab* file.

You can use the **fstyp(1M)** command to determine the type of a file system (for example, if the *vfstab* file contains incorrect information).

To check the file system type, enter:

```
fstyp special
```

## Making (Creating) File Systems

---

### Why/When

Common occasions for creating file systems include:

- To add a new disk
- To create or change slices in the UNIX partition on a disk
- To recover disk information after replacing the disk
- To store data outside the *root* file system

If you add a disk using the menu interface, **mkfs** is invoked automatically to create any file systems you specify.

**NOTE:** If you create file systems using the menu interface (**sysadm** or **OSA**), you can not change the default values for the logical block sizes. If you wish to create a file system with a logical block size other than the default values which the menu interfaces use, you must re-create the file system with the command line.

If a disk is already formatted, partitioned, and sliced, you can use the **mkfs(1M)** command or **sysadm** menus to create a new file system in any of the slices.

---

### Before You Start

- If the slice already contains a file system, be sure you back up the existing data before creating a new file system in the slice.

- Determine the logical block size for the file system. Different file system types have different logical block sizes. Figure 4–1 shows possible logical block sizes for s5 or ufs file systems.

| If you use . . . | The file system block size can be . . .<br>(bytes/block) |
|------------------|----------------------------------------------------------|
| s5               | 512, 1024, or 2048                                       |
| ufs              | 4096 or 8192                                             |

Figure 4–1: Possible File System Block Sizes

**NOTE:** Different defaults are used for the logical block sizes depending on whether you use the menu interfaces (sysadm or OSA) to create file systems or the command line. A ufs file system created using the command line uses a default value of 8K for the logical block size while it uses a default value of 4K when it is created through the menu interfaces. An s5 file system created through menu interfaces or the command line always uses a default of 1024 bytes.

The default logical block size generally gives the best performance for an s5 file system.

For a ufs file system, the default logical block size for the root file system is 4K (because it is created at installation time using the menu interface) while the default for other ufs file systems is 8K when they are created using the command line. You may find a 4K block size gives better performance unless all files in a file system are extremely large. Regardless of which logical block size you choose, a 1K fragment size is generally best.

For an s5 or ufs file system, the **mkfs(1M)** (see also **mkfs\_s5(1M)** and **mkfs\_ufs(1M)** for file system specific options) command creates a file system with a *root* directory and a *lost+found* directory.

Before creating a file system, use the **prtvtoc(1M)** command to display the number of blocks allocated for each slice of the given disk. Record the value from the *length* field of the slice on which you want to create your file system.

## Procedure

At the # prompt, specify the **mkfs(1M)** command with appropriate options.

The syntax of the **mkfs** command is:

```
mkfs [-F] [-V] [-m] [current_options] \
[-o specific_options] special [operands]
```

The **mkfs** command automatically creates the *lost+found* directory.

**NOTE:** If OA&M or OSA is installed, run **mktable** to update the object table with the new file system:

```
/usr/sadm/sysadm/bin/mktable &
```

## Examples

The following entry creates an **s5** file system with a logical block size of 1K and a size of 20000 512-byte blocks. The file system is created in slice 4 of the internal hard disk in the first slot of an NCR 3550 computer.

```
mkfs -F s5 /dev/rdisk/c0t0d0s4 20000
```

The following entry creates a **ufs** file system with a logical block size of 4K and a size of 41000 512-byte blocks. The file system is created in slice 4 of the internal hard disk in the second slot of an NCR 3550 computer.

```
mkfs -F ufs -o bsize=4096 /dev/rdisk/c0t1d0s4 \
41000
```

After creating a ufs file system, record and save some of the superblock numbers which the **mkfs(1M)** command displays. These numbers can be used during file system recovery. See “Checking and Repairing ufs File Systems” for more information about file system recovery.

---

### References

Refer to **mkfs(1M)** or **mkfs\_FSType(1M)** in the *Reference Manual* for more information about command syntax and usage.

# Mounting File Systems

---

## Why/When

Before you can access or use a file system, you must establish a connection between the device and the file system name. Making this connection is called mounting the file system.

The boot procedure always mounts the *root* (and */usr* if separate) file system. When you create a new file system through the menu interface, you can specify that it be mounted automatically as the system enters multiuser mode.

---

## Before You Start

- Be sure the directory on which you plan to mount the file system already exists; otherwise you can not mount the file system.
- Determine the device name for the system you are mounting. Device names are located in subdirectories of the */dev* directory. Refer to Appendix B for more information about device names.
- Determine what file systems are currently mounted and at what directory they are mounted by entering **mount** (with no options) at the system prompt.

Remember: you can not change (**cd**) to a directory in the file system you are mounting before you actually mount the file system, nor can you access files and directories that were previously available below the mount point directory while a file system is mounted there.



---

## Procedure

Enter the appropriate command at the # prompt:

- **mountall(1M)** — Multiple file systems. This command checks the file system table (default */etc/mnttab*) to determine what file systems to mount.
- **mount(1M)** — Only one file system.

If you create a file system from the command line, you can place an entry for the file system in the */etc/mnttab* file to cause the file system to be mounted automatically.

---

## Examples

To mount the ufs file system that physically resides in slice 4 of the internal hard disk in the first slot of an NCR 3550 computer on the directory */mnt*, enter:

```
mount -F ufs /dev/dsk/c0t0d0s4 /mnt
```

To mount the cdfs file system that physically resides on the CD-ROM disk on the directory */cdrom*, using file system specific options, enter:

```
mount -F cdfs -ol,m /dev/dsk/c0t5d0s0 /cdrom
```

**NOTE:** The cdfs file system is a read-only file system and mount is the **ONLY** command that can be used for this file system type. For a description and use of the cdfs file system type refer to Appendix A. You can only mount a cdfs file system when the cdfs package is installed. This package is included on the base operating system install tape and can be installed separately.

---

## References

Refer to **mountall(1M)**, **mount(1M)** and **mount\_FSType(1M)** in the *Reference Manual* for information about command syntax and usage.

Refer to Appendix A for more information about file systems.

# Unmounting File Systems

---

## Why/When

You may want to unmount a file system to:

- Check the file system or use other commands that operate on an unmounted file system.
- Access files and directories under the mount point that are not available when the file system is mounted.

---

## Before You Start

- Be sure no one is using the file system(s) you plan to unmount.
- If you are only unmounting a single file system, determine the name of the directory on which the file system is mounted.

---

## Procedure

Enter the appropriate command at the # prompt:

- `umountall(1M)` — All mounted file systems except *root*.
- `umount(1M)` — A single, specified file system.

---

### Example

To unmount the file system that is currently mounted on the directory */mnt*, enter:

```
umount /mnt
```

---

### References

Refer to “Mounting File Systems” for information on mounting a file system.

Refer to **umount(1M)** and **umountall(1M)** in the *Reference Manual* for information about command syntax and usage.

# Displaying File System Information

## Why/When

Use this task to display the mount point, partition names, size, and number of blocks for a specific file system.

## Before You Start

Be sure the file system you want information about is mounted.

## Procedure

Enter the following command:

```
df -g
```

## Example

For each mounted file system, the display is similar to the following:

```
/mnt (/dev/dsk/c0t0d0s3): 8192 block size 1024 frag size
2774 total blocks 2756 free blocks 2748 available 512 total files
508 free files 9175043 filesys id
ufs fstype 0x00000004 flag 255 filename length
```

This display shows the following information:

### Mount Point

The name of the directory on which the file system is mounted

**Partition Name**

The device special name of the partition in which the file system resides

**Logical Block Size**

The number of bytes in a logical block of the file system

**Fragment Size**

The number of bytes that is considered a “fragment” of a logical block for the file system (only applies to the ufs file system type)

**Total Blocks**

The total number of 512–byte blocks in the file system (this number is slightly less than the number of 512–byte blocks specified as the size of the file system when it was created)

**Free Blocks**

The number of 512–byte blocks in the file system that are not in use

**Available Blocks**

The number of 512–byte blocks in the file system that are available for data blocks

**Total Files**

The total number of files that can be created in the file system at any given time (in an s5 or ufs type file system, this is the number of inodes)

**Free Files**

The number of files that can still be created in the file system

**Filesystem ID**

A unique number identifying the file system

**File System Type**

The type of file system (s5, ufs, bfs, etc.)

**Flag**

A unique number used by the system to identify certain permissions and file system characteristics

**Filename Length**

The maximum length permitted for a file name

---

**References**

Refer to **df(1M)** or **df\_FSType(1M)** in the *Reference Manual* for more information about command syntax and usage.

## Compacting Files in */stand*

The */stand* file system is a small file system with contiguous blocks, and is designed to be used only for booting. For this reason, you should not copy or create files in */stand*.

---

### Why/When

If you copy */stand/unix* to */stand/unix.save* before you create another kernel, and you receive an error message indicating that there is not enough space available, you might need to compact the */stand* file system. Check the total free space on */stand* with the command `df -t` and compare that with the size of the file you are trying to copy to */stand*. If the numbers indicate that there is enough room, then you need to compact the */stand* file system. The file is only copied successfully if all blocks can be allocated contiguously; that is, the file can not be fragmented.

Normally, the system compacts */stand* automatically to keep the blocks contiguous and the free space usable. If your system indicates there is not room in */stand* for the action it is trying to perform AND the system does not automatically compact the file system after a reboot, you can use the following procedure to perform the compacting.

**NOTE:** A message is displayed on the screen whenever the system is compacting */stand*.

---

## Procedure

Step 1: Copy all files from */stand* to a temporary directory that is NOT in */tmp*. For example:

```
cp /stand/* /mytmp
```

Step 2: Perform the following command:

```
mkfs -F bfs /dev/dsk/device_name 20480
```

The *device\_name* is the name of slice containing */stand*. See Appendix B, “Device Names and Numbers,” for an explanation of device names.

Step 3: Copy the files, one at a time, from */mytmp* to */stand*.

Step 4: Perform the following commands:

```
ln -s /stand/unix /unix
sync
```



## Using Quotas

---

### Why/When

To limit the two principal file system resources (inodes and data blocks) for each user, you can assign a user different quotas for each different ufs file system. Quotas for a user consist of:

- Hard limits (one for inodes and one for blocks) that represent an absolute limit on the resource, a limit the user can never exceed.
- Soft limits (one for inodes and one for blocks) that represent the normal limit on the resource.
- Time limits (one for inodes and one for blocks) that specify the length of time users can exceed their soft limit. The time limits apply to all users of the file system; they are not assigned for each user.

---

### Procedures

Limiting file system resources involves two phases: establishing quotas and managing quotas.

#### Establishing Quotas

Before turning on quotas for the first time, establish quotas by doing the following:

- Step 1: If the quotas are for a file system that is listed in */etc/vfstab*, enter **rw,quota** in the *mntopts* field for the file system.

- Step 2: Mount the file system and change directories (`cd`) to the mount point.
- Step 3: In the root directory for the file system, create a file called *quotas* that is owned by `root` and not writable by others.
- Step 4: To change the time limits for exceeding block and/or inode soft limits, enter the following command:

```
edquota -t
```

This command places you in a file that specifies the time limits and permits you to edit the limits using the `vi` editor.

Initially, the time limits are set to the values defined for `DQ_FTIMELIMIT` and `DQ_BTIMELIMIT` in `/usr/include/sys/fs/ufs_quota.h` (normally, one week). If you leave any limit set at 0, these default values are used.

- Step 5: To set quotas for a user, enter the `edquota` command (with or without the `-p` option). Once you set the quotas for one user, you can invoke `edquota` with the `-p` option to set the same quotas for other users, even users not yet added to your system.

## Managing Quotas

After quotas are established, you can manage them using the following procedure:

- Step 1: Before turning on quotas for a file system, run `quotacheck` on the file system.

This command updates quotas to reflect the current state (in case the file system has been used since the last time quotas were turned on) and provides a sanity check on the *quotas* file.

**Step 2:** Run **quotaon** to turn quotas on. Use the **-a** option to turn quotas on for each ufs file system rather than just for one that you specify.

**Step 3:** Obtain quota information for users when desired:

- To obtain quota information for all users on a file system, use the **repquota** or **quota** command.
- To obtain quota information on a single user, use the **quota** command.

Users can obtain their own quota information using the **quota** command.

**Step 4:** Run **quotaoff** to turn quotas off. Use the **-a** option to turn quotas off for every ufs file system rather than just for one that you specify.

---

### Additional Information

If a user exceeds his or her soft limit for blocks or inodes, a timer is started. No error occurs as long as the user reduces the use of that resource below his or her soft limit before the time limit expires. If the time limit for that resource expires while the user is still exceeding the soft limit, the user receives error messages saying that the file system is full. These messages persist until the user reduces usage to a level below the soft limit. If a user tries to exceed the hard limit at any time, the attempt fails, indicating there is no more space in the file system.

Since users receive no warning when they exceed their soft limit, they should run **quota(1)** frequently. Additionally, users should include **quota** in their *.profile* so that it runs when they log in.

# **Checking and Repairing the File System**

This chapter includes information about the following tasks and problems:

|                                                      |      |
|------------------------------------------------------|------|
| Minimizing File System Corruption .....              | 5-2  |
| Using fsck to Check and Repair the File System ..... | 5-4  |
| Checking and Repairing s5 File Systems .....         | 5-8  |
| s5 File System Components Checked by fsck .....      | 5-11 |
| Phases of fsck on s5 File Systems .....              | 5-19 |
| Checking and Repairing ufs File Systems .....        | 5-38 |
| ufs File System Components Checked by fsck .....     | 5-40 |
| Phases of fsck on ufs File Systems .....             | 5-47 |
| Checking bfs File Systems .....                      | 5-78 |

**NOTE:** If you have the vxfs package (Journaling File System) on your system, refer to the *Journaling File System Administrator Guide* for the file system specific fsck options.

## Minimizing File System Corruption

A file system can be corrupted by hardware failure, program interrupts, human error, or a combination of these. This section describes why corruption occurs, how to minimize corruption, and what to do if it does occur.

---

### Why Corruption Occurs

The system maintains a copy of the following file system structures in memory for fast access, updating them to disk periodically:

- Each mounted file system's super block (including the root (/) file system)
- Each active inode
- Each data block that is being referenced (the buffer area)

File system inconsistencies can occur if the system halts. If the system halts, only the disk version is available. The corruption occurs because the disk version of the super blocks, the inodes, and data blocks do not match the memory version.

---

### How To Minimize Corruption

#### How Does the System Minimize File System Corruption?

By default, an automatic **sync** feature causes the memory version of modified super blocks, modified inodes, and delayed data blocks to be written to disk every minute. To change the **sync** rate, modify the **NAUTOUP** parameter. In addition, you can invoke the **sync** command or system call to perform the same action.

When a file is closed, its inode and the data blocks that are in memory are automatically updated to disk. When a data block in the buffer area is filled (last byte written), the block is updated to disk.

### **How Can You Minimize File System Corruption?**

To help prevent file system corruption:

- NEVER remove a file system physically without first unmounting it.
- NEVER physically write-protect a mounted file system, unless it is mounted as read-only.
- Use the `fsck(1M)` utility at least once a week to check the integrity of ufs, s5, and bfs file systems. Enter single user mode and then perform the check. For the root file system, follow the procedure in the “Changing Run Levels” section of Chapter 1, “Being a Systems Administrator.”

### **How Can Users Minimize File System Corruption?**

Users should back up their important files on a frequent basis so that they can recover them if file system problems occur. The frequency of backups depends on the importance and the amount of the data.

## Using fsck to Check and Repair the File System

The **fsck(1M)** utility is an interactive file system check and repair command that acts on a ufs, s5 or bfs type file system. It reports any problems or inconsistencies in the file system. You may choose to have **fsck** fix any problems or you may decide to fix the problems manually.

**NOTE:** Any corrections are made to the DISK version of the super block.

If your system software detects file system corruption during system startup and your system cannot go into multiuser mode, follow the procedure in the “Changing Run Levels” section of Chapter 1, “Being a Systems Administrator.”

---

### Why/When

Use the **fsck** utility:

- To check the file system to ensure that all files are referenced correctly, the ilist is in order, and the super block is current.
- To repair the file system before attempting to use your system if you encounter problems during a file system check.
- To check the file system at least once a week for every system and daily for a heavily used system.

---

### Before You Start

You should do the following:

- Review this entire chapter. (Later sections of this chapter describe the format, options, and phases of **fsck**.)
- Put the system in single user mode before checking the *root (/)* file system. If normal file system activity occurs while a file system is being checked, **fsck** may find errors when no errors exist. This false error detection occurs because the file system is changing as **fsck** is checking. To prevent activity while checking, you can also check the *root (/)* file system using the Maintenance file system (*/install*).
- Unmount removable file systems before checking them.

---

### Procedure

Using **fsck** minimizes the risk of losing actual data. By using **fsck** more often, you learn to recognize what responses do not affect valid data (for example, clearing an inode for a file of size 0).

### Checking the File System

Step 1: Run **fsck** with the **-n** option or enter no in response to all questions. Make a note of all errors.

Step 2: Use **ncheck(1M)** to determine the file names associated with any inode listed as an error (if the file name is not given by **fsck**).

See **ncheck(1M)** in the Reference Manual for more information.

Step 3: If the files are not backed up, attempt to back them up to preserve as much data as possible.

See the “Backup and Restore” chapter.

### Correcting the File System

Step 1: Run **fsck** again to correct the errors and set the file system state to OKAY.



- Step 2: If you are repairing the *root* file system, reboot the system after the repairs are made. If you are repairing a removable file system, mount the file system after repairs are made.
- Step 3: Examine the files in the *lost+found* directory.
- Determine whether they are still needed and, if so, to what directory or file name they should be assigned.
- Step 4: Restore, from the backup if possible, any files cleared or removed by **fsck**.

---

### Generic Format of fsck

You can use **fsck** to check ufs, s5 or bfs file systems, but the format of **fsck** is slightly different for each one. This section discusses the general format of **fsck**. Following sections discuss the specific format of **fsck** for each type of file system.

```
fsck [-F fstype] [-V] [-m] [special . . .]
```

```
fsck [-F fstype] [-V] [current_options]
[special...] [-o specific_options] [special...]
```

#### **-F *fstype***

Specifies the file system type on which to operate. *fstype* must be specified on the command line or must be determinable from */etc/vfstab* by matching an entry in that file with the *special* specified.

#### **-V**

Echoes the command line, including additional information derived from */etc/vfstab*. This option verifies and validates the command line but does not cause the command to execute.

#### **-m**

Performs a sanity check only. This option is usually used before mounting file systems because it lets the administrator know whether the file system needs to be checked.

### *current\_options*

Specify options supported by the *s5*-specific module of the command.

### **-o** *specific\_options*

Specify *fstype*-specific options, if any. *specific\_options* are specified in a comma-separated list of keywords and/or keyword-attribute pairs for interpretation by the *fstype*-specific module of the command.

If the file system is inconsistent, **fsck** prompts you for concurrence before it attempts each correction. Some corrective actions result in some loss of data. You can determine the amount and severity of data loss from the diagnostic output. By default, **fsck** waits for you to respond yes or no before making any correction. If you do not have write permission, **fsck** defaults to a no action.

## Checking and Repairing s5 File Systems

This section describes the format and options of the **fsck** command for s5 file systems. For an explanation of the components checked for an s5 file system, see the section entitled “s5 Components Checked by **fsck**”. For explanations of error messages from **fsck**, see the following section “Phases of **fsck** on s5 File Systems”.

---

### Format of **fsck** for s5 File Systems

The following is the s5 specific format of the **fsck** command:

```
fsck [-F s5] [generic_options] [special...]

fsck [-F s5] [generic_options] [-y] [-n] [-p] \
[-sX] [-SX] [-tfile] [-l] [-q] [-D] [-f] \ [special...]
```

(The second command line is shown on multiple lines for readability.)

### Options

The options are as follows:

**-y**

Specifies a **yes** response for all questions. This is the normal choice when the command is being run as part of a shell procedure. **fsck** corrects all errors.

**-n**

Specifies a **no** response for all questions. **fsck** does not write to the file system.

- p**  
Corrects any innocuous inconsistencies. Unreferenced blocks, misaligned directories, incorrect link counts and bad free lists are some examples of inconsistencies which are automatically corrected.
- sX**  
Specifies an unconditional reconstruction of the free list. The *X* argument specifies the number of blocks-per-cylinder and the number of blocks to skip (rotational gap). The default values are those specified when the file system was created.
- SX**  
Specifies a conditional reconstruction of the free list, to be done only if corruption is detected. The format of the *X* argument is the same as described above for the **-s** option.
- tfile**  
Specifies a scratch file for use in case the file system check requires additional memory. If this option is not specified, the process asks for a filename when it needs more memory.
- l**  
Identifies damaged files by their logical names in addition to the inode numbers.
- q**  
Specifies a quiet file system check. Output messages from the process are suppressed.
- D**  
Checks directories for bad blocks. This option is used to check file systems for damage after a system crash.

## Checking and Repairing the File System

**-f**

Specifies a fast file system check. Only Phase 1 (check blocks and sizes) and Phase 5 (check free list) are executed for a fast check. Phase 6 (reconstruct free list) is run only if necessary.

*special*

Names the special device file associated with a file system. If no device name is specified, **fsck** checks all file systems named in */etc/vfstab* with a numeric *fsckpass* field.

### Sample Command Line

In this example the */usr* file system is located in slice */dev/rdsk/c0t6d0s3*. The following command line checks the */usr* file system with no options:

```
fsck -F s5 /dev/rdsk/c0t6d0s3
```

## s5 File System Components Checked by fsck

This section discusses the components of an s5 file system and describes the kinds of consistency checks that are applied to each.

**fsck** checks the following components:

- Super-block
- Inodes
- Indirect blocks
- Directory data blocks
- Regular data blocks

---

### Super-Block

Every change to the file system blocks or inodes modifies the super-block. If the CPU is halted and the last command involving output to the file system is not a **sync** command, the super-block will almost certainly be corrupted. The super-block can be checked for inconsistencies involving:

- File system size
- Inode list size
- Free-block list
- Free-block count
- Free inode count

### File System Size and I-Node List Size

The number of blocks in a file system must be greater than the number of blocks used by the super-block plus the number of blocks used by the inode list. The number of inodes must be less than the maximum number allowed for the file system type. While there is no way to check these sizes precisely, **fsck** can check that they are within reasonable bounds. All other checks of the file system depend on the reasonableness of these values.

### Free-Block List

The free-block list starts in the super-block and continues through the free-list blocks of the file system. Each free-list block can be checked for the following:

- List count out of range
- Block numbers out of range
- Blocks already allocated within the file system

**fsck** checks to see that all the blocks in the file system were found.

The first free-block list is in the super-block. **fsck** checks the list count for a value less than 0 or greater than 50. It also checks each block number to make sure it is within the range bounded by the first and last data block in the file system. Each block number is compared to a list of previously allocated blocks. If the free-list block pointer is not 0, the next free-list block is read and the process is repeated.

When all the blocks have been accounted for, **fsck** checks to see if the number of blocks in the free-block list plus the number of blocks claimed by the inodes equals the total number of blocks in the file system. If anything is wrong with the free-block list, **fsck** can rebuild it, leaving out blocks already allocated.

### **Free–Block Count**

The super–block contains a count of the total number of free blocks within the file system. **fsck** compares this count to the number of blocks it finds free within the file system. If the counts do not agree, **fsck** can replace the count in the super–block by the actual free–block count.

### **Free Inode Count**

The super–block contains a count of the number of free inodes within the file system. **fsck** compares this count to the number of inodes it found free within the file system. If the counts do not agree, **fsck** can replace the count in the super–block by the actual free inode count.

---

## **Inodes**

The list of inodes is checked sequentially starting with inode 1 (there is no inode 0). Each inode is checked for inconsistencies involving:

- Format and type
- Link count
- Duplicate blocks
- Bad block numbers
- Inode size



### Format and Type

Each inode contains a mode word. This mode word describes the type and state of the inode. Inodes may be one of six types:

- Regular
- Directory
- Block special
- Character special
- FIFO (named-pipe)
- Symbolic link

Inodes may be in one of three states: unallocated, allocated, and partially allocated. If an inode is partially allocated, the inode is incorrectly formatted. An inode can get into this state if, for example, bad data are written into the inode list because of a hardware failure. The only corrective action **fsck** can take in this situation is to clear the inode.

### Link Count

Each inode contains a count of the number of directory entries linked to it. **fsck** verifies the link count of each inode by examining the entire directory structure, starting from the root directory, and calculating an actual link count for each inode.

Discrepancies between the link count stored in the inode and the actual link count as determined by **fsck** may be of three types:

- The stored count is not 0, the actual count is 0

No directory entry appears for the inode. **fsck** can link the disconnected file to the *lost+found* directory.

- The stored count is not 0, the actual count is not 0, but the counts are unequal

A directory entry has been removed but the inode has not been updated. **fsck** can replace the stored link count by the actual link count.

- The stored count is 0, the actual count is not 0

**fsck** can change the link count of inode to the actual count.

### Duplicate Blocks

Each inode contains a list of all the blocks claimed by the inode. **fsck** compares each block number claimed by an inode to a list of allocated blocks. If a block number claimed by an inode is on the list of allocated blocks, it is put on a list of duplicate blocks. If the block number is not on the list of allocated blocks, it is put on it.

If this process produces a list of duplicate blocks, **fsck** makes a second pass of the inode list to find the other inode that claims each duplicate block. (A large number of duplicate blocks in an inode may be caused by an indirect block not being written to the file system.) Although it is not possible to determine with certainty which inode is in error, in most cases the inode with the most recent modification time is correct. The **fsck** program prompts you to clear both inodes.

### Bad Block Numbers

**fsck** checks each block number claimed by an inode to see that its value is higher than that of the first data block and lower than that of the last block in the file system. If the block number is outside this range, it is considered a bad block number.

Bad block numbers in an inode may be caused by an indirect block not being written to the file system. **fsck** prompts you to clear the inode.

**NOTE:** A bad block number in a file system is not the same as a bad (that is, unreadable) block on a hard disk.

### Inode Size

Each inode contains a 32-bit (4-byte) size field. This field shows the number of characters in the file associated with the inode. A directory inode within the file system has the directory bit set in the inode mode word.

If the directory size is not a multiple of 16, **fsck** warns of directory misalignment and prompts for corrective action.

For a regular file, **fsck** can perform a rough check of the consistency of the size field of an inode. **fsck** uses the number of characters shown in the size field to calculate how many blocks should be associated with the inode, and compares that to the actual number of blocks claimed by the inode.

---

### Indirect Blocks

Indirect blocks are owned by an inode. Therefore, inconsistencies in an indirect block directly affect the inode that owns it. Inconsistencies that can be checked are:

- Blocks already claimed by another inode
- Block numbers outside the range of the file system

The consistency checks for direct blocks described in the sections “Duplicate Blocks” and “Bad Block Numbers” are also performed for indirect blocks.

---

### Directory Data Blocks

Directories are distinguished from regular files by an entry in the *mode* field of the inode. Data blocks associated with a directory contain the directory entries. **fsck** checks directory data blocks for inconsistencies involving:

- Directory inode numbers pointing to unallocated inodes
- Directory inode numbers greater than the number of inodes in the file system

- Incorrect directory inode numbers for “.” and “..” directories
- Directories disconnected from the file system

### Directory Unallocated

If a directory entry inode number points to an unallocated inode, **fsck** can remove that directory entry. This condition occurs if the data blocks containing the directory entries are modified and written out while the inode is not yet written out.

### Bad Inode Number

If a directory entry inode number is pointing beyond the end of the inode list, **fsck** can remove that directory entry. This condition occurs if bad data are written into a directory data block.

### Incorrect “.” and “..” Entries

The directory inode number entry for “.” should be the first entry in the directory data block. Its value should be equal to the inode number for the directory data block.

The directory inode number entry for “..” should be the second entry in the directory data block. Its value should be equal to the inode number for the parent of the directory entry (or the inode number of the directory data block if the directory is the root directory). If the directory inode numbers for “.” and “..” are incorrect, **fsck** can replace them with the correct values.

### Disconnected Directories

The **fsck** program checks the general connectivity of the file system. If it finds a directory that is not linked to the file system, **fsck** links the directory to the *lost+found* directory of the file system. (This condition can occur when inodes are written to the file system but the corresponding directory data blocks are not.) When a file is linked to the *lost+found* directory, the owner of the file must be notified.

---

### **Regular Data Blocks**

Data blocks associated with a regular file hold the file's contents. **fsck** does not attempt to check the validity of the contents of a regular file's data blocks.

## Phases of fsck on s5 File Systems

This section describes the phases of **fsck**, the messages you may see, and possible responses to these messages.

---

### fsck Passes

**fsck** is a multi-pass file system check program. Each file system pass invokes a different phase of the **fsck** program. After initialization, **fsck** performs successive passes over each file system, checking blocks and sizes, path names, connectivity, reference counts, and the map of free blocks (possibly rebuilding it), and performs some cleanup.

---

### fsck Phases

The **fsck** program runs in six phases with each phase reporting any errors that it detects. If the error is one that **fsck** can correct, the system asks if the correction should be made. This section describes the messages that are produced by each phase.

For example, the command line below checks the */usr* file system with no options. As each phase is completed, a message appears. In this case, no inconsistencies were detected. At the end of the program a summary message shows the number of files (inodes) used, blocks used, and free blocks.

```
fsck -F s5 /dev/rdisk/clt6d0s2
/dev/rdisk/clt6d0s2
File System: usr Volume: usr

** Phase 1 - Check Blocks and Sizes
** Phase 2 - Check Pathnames
** Phase 3 - Check Connectivity
** Phase 4 - Check Reference Counts
** Phase 5 - Check Free List
289 files 6522 blocks 3220 free
```

The following abbreviations appear in the messages:

- **BLK** — block number
- **DUP** — duplicate block
- **DIR** — directory name
- **MTIME** — time file was last modified
- **UNREF** — unreferenced
- **CG** — cylinder group

The following single-letter abbreviations are replaced by the text when the message appears on your screen:

- **B** — block number
- **F** — file (or directory) name
- **I** — inode number
- **M** — file mode
- **O** — user-id of a file's owner
- **S** — file size
- **T** — time file was last modified
- **X** — link count, or number of **BAD**, **DUP**, or **MISSING** blocks, or number of files (depending on context)
- **Y** — corrected link count number or number of blocks in file system (depending on context)
- **Z** — number of free blocks

---

## Initialization Phase

In this phase, **fsck** checks command line syntax, sets up some tables, and opens some files. The **fsck** program terminates when it encounters errors during the initialization phase.

### General Errors

The following three error messages may appear in any phase after initialization. While they offer the option to continue, it is generally best to regard them as fatal, end the run, and try to determine what caused the problem.

#### **CAN NOT SEEK: BLK B (CONTINUE?)**

A request to move a specified block number *B* in the file system failed. This message indicates a serious problem, probably a hardware failure.

#### **CAN NOT READ: BLK B (CONTINUE?)**

A request to read a specified block number *B* in the file system failed. This message indicates a serious problem, probably a hardware failure.

#### **CAN NOT WRITE: BLK B (CONTINUE?)**

A request to write a specified block number *B* in the file system failed. The disk may be write-protected.

### Responses

At the **CONTINUE** prompt, enter **no** to terminate the program.

These errors are not always fixed by the first pass of **fsck**. If any of these errors occurs, run **fsck** again until you do not get the error message.



### **Phase 1: Check Blocks and Sizes**

This phase checks the inode list. It reports error conditions encountered while:

- Checking inode types
- Setting up the zero-link-count table
- Examining inode block numbers for bad or duplicate blocks
- Checking inode size
- Checking inode format

### **Types of Error Messages – Phase 1**

Phase 1 produces four types of error messages:

- Informational messages
- Messages with a CONTINUE? prompt
- Messages with a CLEAR? prompt
- Messages with a RECOVER? prompt

There is a connection between some informational messages and messages with a CONTINUE? prompt. The CONTINUE? prompt generally indicates that some limit has been reached.

### **Responses to Error Messages**

Possible responses to the CONTINUE prompt are:

- YES — Continue with the program
- NO — Terminate the program

When this error occurs, a complete check of the file system is not possible. Run `fsck` a second time to recheck the file system.

Possible responses to the RECOVER prompt are:

- YES — Continue with the program.
- NO — Recover all the blocks to which the inode points.

A no response is only appropriate if you intend to delete the excess blocks.

Possible responses to the CLEAR prompt are:

- YES — Deallocate the inode *I* by zeroing out its contents. This may generate the UNALLOCATED error condition in Phase 2 for each directory entry pointing to this inode.
- NO — Ignore the error condition.

### Phase 1 Error Messages

#### UNKNOWN FILE TYPE I= I (CLEAR?)

The mode word of the inode *I* indicates that the inode is not a pipe, special character inode, regular inode, or directory inode. If you specified the **-p** option, the inode is cleared.

#### LINK COUNT TABLE OVERFLOW (CONTINUE?)

An internal table for **fsck** containing allocated inodes with a link count of zero has no more room. If you specified the **-p** option, the program exits and you must complete **fsck** manually.

#### B BAD I= I

Inode *I* contains block number *B* with a number lower than the number of the first data block in the file system or greater than the number of the last block in the file system. This error condition may generate the EXCESSIVE BAD BLKS error message in Phase 1 if inode *I* has too many block numbers outside the file system range. This error condition generates the BAD/DUP error message in Phases 2 and 4.

**EXCESSIVE BAD BLOCKS I= I (CONTINUE?)**

Too many (usually more than 10) blocks have a number lower than the number of the first data block in the file system or greater than the number of the last block in the file system associated with inode *I*. If you specified the **-p** option, the program terminates.

**B DUP I= I**

Inode *I* contains block number *B*, which is already claimed by the same or another inode or by a free-list. This error condition may generate the EXCESSIVE DUP BLKS error message in Phase 1 if inode *I* has too many block numbers claimed by the same or another inode or by a free-list. This error condition invokes Phase 1B and generates the BAD/DUP error messages in Phases 2 and 4.

**EXCESSIVE DUP BLKS I= I (CONTINUE?)**

Too many (usually more than 10) blocks are claimed by the same or another inode or by a free-list. If you specified the **-p** option, the program terminates.

**DUP TABLE OVERFLOW (CONTINUE?)**

An internal table in **fsck** containing duplicate block numbers has no more room. If you specified the **-p** option, the program terminates.

**DIRECTORY MISALIGNED I= I**

The size of a directory inode is not a multiple of 16. If you specified the **-p** option, the directory is recovered automatically.

**PARTIALLY ALLOCATED INODE I= I (CLEAR?)**

Inode *I* is neither allocated nor unallocated. If you specified the **-p** option, the inode is cleared.

**DIR/FILE SIZE ERROR**

The file references more or less data than is indicated by the inode.

### **DELETE OR RECOVER EXCESS DATA**

You can choose to delete or to recover the excess blocks pointed to by the inode.

#### **RECOVER?**

The file references more data than is indicated by the inode. You can choose to correct the inode information. If you specified the **-p** option, the data is recovered.

#### **DELETE?**

The file references more data than is indicated by the inode. You can choose to delete the referenced blocks and leave the inode data intact.

---

### **Phase 1B: Rescan for More DUPS**

When **fsck** finds a duplicate block in the file system, it rescans the file system to find the inode that previously claimed that block. When it finds the duplicate block, it prints the following informational message:

#### **DUP I= I**

Inode *I* contains block number *B* that is already claimed by the same or another inode or by a free-list. This error condition generates the BAD/DUP error message in Phase 2. To determine inodes that have overlapping blocks, examine this error condition and the DUP error condition in Phase 1.

---

### **Phase 2: Check Pathnames**

This phase removes directory entries pointing to bad inodes found in Phases 1 and 1B. It reports error conditions resulting from:

- Incorrect root inode mode and status
- Directory inode pointers out of range
- Directory entries pointing to bad inodes

### Types of Error Messages—Phase 2

Phase 2 has four types of error messages:

- Informational messages
- Messages with a **FIX?** prompt
- Messages with a **CONTINUE?** prompt
- Messages with a **REMOVE?** prompt

### Responses

Possible responses to the **FIX?** prompt are:

- **YES** — Change the root inode type to “directory.”
- **NO** — Terminate the program because **fsck** will be unable to continue.

If the root inode data blocks are not directory blocks, a very large number of error messages are generated.

Possible responses to the **CONTINUE?** prompt are:

- **YES** — Ignore the **DUPS/BAD IN ROOT INODE** error message and continue to run the file system check.
- **NO** — Terminate the program.

If the root inode is not correct, a large number of other error messages may be generated.

Possible responses to the **REMOVE?** prompt are:

- **YES** — Remove duplicate or unallocated blocks.
- **NO** — Ignore the error condition. A no response is only appropriate if you intend to take other measures to fix the problem.

## Phase 2 Error Messages

### ROOT INODE UNALLOCATED. TERMINATING

The root inode (usually inode number 2) of the file system has no allocate mode bits. This error message indicates a serious problem that causes the program to stop. Call your service representative.

### ROOT INODE NOT DIRECTORY (FIX?)

The root inode (usually inode number 2) of the file system is not directory inode type. If you specified the **-p** option, the program terminates.

### DUPS/BAD IN ROOT INODE (CONTINUE?)

Phase 1 or 1B found duplicate blocks or bad blocks in the root inode (usually inode number 2) of the file system. If you specified the **-p** option, the program terminates.

### I OUT OF RANGE I= I NAME= F (REMOVE?)

A directory entry *F* has an inode number *I* that is greater than the end of the inode list. If you specified the **-p** option, the inode is removed automatically.

### UNALLOCATED I= I OWNER= O MODE= M SIZE= S MTIME= T NAME= F (REMOVE?)

A directory entry *F* has an inode *I* without allocate mode bits. The owner *O*, mode *M*, size *S*, modify time *T*, and filename *F* are printed. If the file system is not mounted and you did not specify the **-n** option, the entry is removed automatically if the inode it points to is character size 0. The entry is removed if you specified the **-p** option.

**DUP/BAD I= I OWNER= O MODE= M SIZE= S MTIME= T  
DIR= F (REMOVE?)**

Phase 1 or Phase 1B found duplicate blocks or bad blocks associated with directory entry *F*, directory inode *I*. The owner *O*, mode *M*, size *S*, modify time *T*, and directory name *F* are printed. If you specified the **-p** option, the duplicate/bad blocks are removed.

**DUP/BAD I= I OWNER= O MODE= M SIZE= S MTIME= T  
FILE=F (REMOVE?)**

Phase 1 or Phase 1B found duplicate blocks or bad blocks associated with file entry *F*, inode *I*. The owner *O*, mode *M*, size *S*, modify time *T*, and filename *F* are printed. If you specified the **-p** option, the duplicate/bad blocks are removed.

**BAD BLK B IN DIR I= I OWNER= O MODE= M SIZE= S  
MTIME=T**

This message only occurs when the **-D** option is used. A physically damaged block was found in directory inode *I*. Error conditions looked for in directory blocks are nonzero padded entries, inconsistent "." and ".." entries, and embedded slashes in the name field. This error message means that you should at a later time either remove the directory inode if the entire block looks bad or change (or remove) those directory entries that look bad.

---

### Phase 3: Check Connectivity

This phase checks the directories examined in Phase 2. It reports error conditions resulting from:

- Unreferenced directories
- Missing or full *lost+found* directories

## Types of Error Messages—Phase 3

Phase 3 has two types of error messages:

- Informational messages
- Messages with a RECONNECT? prompt

## Responses

Possible responses to the RECONNECT? prompt are:

- YES — Reconnect directory inode *I* to the file system in the directory for lost files (usually the *lost+found* directory). This response may generate *lost+found* error messages if there are problems connecting directory inode *I* to the *lost+found* directory. If the link is successful, a CONNECTED informational message appears.
- NO — Ignore the error condition. This response generates UNREF error messages in Phase 4. A no response is appropriate only if you intend to take other measures to fix the problem.

## Phase 3 Error Messages

UNREF DIR I= I OWNER= O MODE= M SIZE= S MTIME= T (RECONNECT?)

The directory inode *I* was not connected to a directory entry when the file system was traversed. The owner *O*, mode *M*, size *S*, and modify time *T* of directory inode *I* are printed. The fsck program forces the reconnection of a non-empty directory. If you specified the *-p* option, the non-empty directory is reconnected.

## SORRY. NO lost+found DIRECTORY

There is no *lost+found* directory in the root directory of the file system; fsck ignores the request to link a directory to the *lost+found* directory. This generates the UNREF error message in Phase 4. The access modes of *lost+found* directory may be incorrect.



### **SORRY. NO SPACE IN *lost+found* DIRECTORY**

There is no space to add another entry to the *lost+found* directory in the root directory of the file system; **fsck** ignores the request to link a directory to the *lost+found* directory. This generates the UNREF error message in Phase 4. Clear out unnecessary entries in the *lost+found* directory or make it larger.

### **DIR I= I1 CONNECTED. PARENT WAS I= I2**

This advisory message indicates that a directory inode I1 was successfully connected to the *lost+found* directory. The parent inode I2 of the directory inode I1 is replaced by the inode number of the *lost+found* directory.

---

## **Phase 4: Check Reference Counts**

This phase checks the link count information obtained in Phases 2 and 3. It reports error conditions resulting from:

- Unreferenced files
- Missing or full *lost+found* directory
- Incorrect link counts for files, directories, or special files
- Unreferenced files and directories
- Bad or duplicate blocks in files and directories
- Incorrect total free-inode counts

### **Types of Error Messages—Phase 4**

Phase 4 has five types of error messages:

- Informational messages
- Messages with a RECONNECT? prompt
- Messages with a CLEAR? prompt
- Messages with an ADJUST? prompt
- Messages with a FIX? prompt

## Responses

Possible responses to the RECONNECT? prompt are:

- YES — Reconnect inode *I* to the file system in the directory for lost files (usually the *lost+found* directory). This response can generate a *lost+found* error message in this phase if there are problems connecting inode *I* to the *lost+found* directory.
- NO — Ignore this error condition. This response generates a CLEAR error message later in Phase 4.

Possible responses to the CLEAR? prompt are:

- YES — Deallocate the inode by zeroing out its contents.
- NO — Ignore the error condition. This response is appropriate only if you intend to take other measures to fix the problem.

Possible responses to the ADJUST? prompt are:

- YES — Replace the link count of file inode *I* by *Y*.
- NO — Ignore the error condition. This response is appropriate only if you intend to take other measures to fix the problem.

Possible responses to the FIX? prompt are:

- YES — Replace the count in super-block by the actual count.
- NO — Ignore the error condition. This response is appropriate only if you intend to take other measures to fix the problem.

### Phase 4 Error Messages

**UNREF FILE I= I OWNER= O MODE= M SIZE= S  
MTIME= T (RECONNECT?)**

Inode *I* was not connected to a directory entry when the file system was traversed. The owner *O*, mode *M*, size *S*, and modify time *T* of inode *I* are printed. If you omitted the **-n** option and the file system is not mounted, empty files are cleared automatically. Non-empty files are not cleared. If you specified the **-p** option, the inode is reconnected.

### **SORRY. NO *lost+found* DIRECTORY**

There is no *lost+found* directory in the root directory of the file system; **fsck** ignores the request to link a file to the *lost+found* directory. This generates the **CLEAR** error message later in Phase 4. The access modes of the *lost+found* directory may be incorrect.

### **SORRY. NO SPACE IN *lost+found* DIRECTORY**

There is no space to add another entry to the *lost+found* directory in the root directory of the file system; **fsck** ignores the request to link a file to the *lost+found* directory. This generates the **CLEAR** error message later in Phase 4. Check the size and contents of the *lost+found* directory.

### **(CLEAR)**

The inode mentioned in the **UNREF** error message immediately preceding cannot be reconnected.

**LINK COUNT FILE I= I OWNER= O MODE= M SIZE= S  
MTIME= T COUNT= X SHOULD BE Y (ADJUST?)**

The link count for file inode *I*, is *X* but should be *Y*. The owner *O*, mode *M*, size *S*, and modify time *T* are printed. If you specified the **-p** option, the link count is adjusted.

**LINK COUNT DIR I= I OWNER= O MODE= M SIZE= S  
MTIME= T COUNT= X SHOULD BE Y (ADJUST?)**

The link count for directory inode *I* is *X* but should be *Y*. The owner *O*, mode *M*, size *S*, and modify time *T* of inode *I* are printed. If you specified the **-p** option, the link count is adjusted.

**UNREF FILE I= I OWNER= O MODE= M SIZE= S  
MTIME= T (CLEAR?)**

File inode *I* was not connected to a directory entry when the file system was traversed. The owner *O*, mode *M*, size *S*, and modify time *T* of inode *I* are printed. If you omitted the **-n** option and the file system is not mounted, empty files are cleared automatically. Non-empty directories are not cleared. If you specified the **-p** option, the file is cleared if it can not be reconnected.

**UNREF DIR I= I OWNER= O MODE= M SIZE= S MTIME=  
T (CLEAR?)**

Directory inode *I* was not connected to a directory entry when the file system was traversed. The owner *O*, mode *M*, size *S*, and modify time *T* of inode *I* are printed. If you omitted the **-n** option and the file system is not mounted, empty directories are cleared automatically. Non-empty directories are not cleared. If you specified the **-p** option, the directory is cleared if it can not be reconnected.

**BAD/DUP FILE I= I OWNER= O MODE= M SIZE= S  
MTIME= T (CLEAR?)**

Phase 1 or Phase 1B found duplicate blocks or bad blocks associated with file inode *I*. The owner *O*, mode *M*, size *S*, and modify time *T* of inode *I* are printed. If you specified the **-p** option, the file is cleared.

**BAD/DUP DIR I= I OWNER= O MODE= M SIZE= S  
MTIME= T (CLEAR?)**

Phase 1 or Phase 1B found duplicate blocks or bad blocks associated with directory inode *I*. The owner *O*, mode *M*, size *S*, and modify time *T* of inode *I* are printed. If you specified the **-p** option, the directory is cleared.

**FREE INODE COUNT WRONG IN SUPERBLK (FIX?)**

The actual count of the free inodes does not match the count in the super-block of the file system. If you specified the **-q** or **-p** option, the count in the super-block is fixed automatically.

---

## **Phase 5: Check Free List**

This phase checks the free-block list. It reports error conditions resulting from:

- Bad blocks in the free-block list
- A bad free-block count
- Duplicate blocks in the free-block list
- Unused blocks from the file system that are not in the free-block list
- An incorrect total free-block count

### **Types of Error Messages – Phase 5**

Phase 5 has four types of error messages:

- Informational messages
- Messages that have a **CONTINUE?** prompt
- Messages that have a **FIX?** prompt
- Messages that have a **SALVAGE?** prompt

## Responses

Possible responses to the CONTINUE? prompt are:

- YES — Ignore the rest of the free-block list and continue running fsck. This response generates the BAD BLKS IN FREE LIST error message later in Phase 5.
- NO — Terminate the program.

Possible responses to the FIX? prompt are:

- YES — Replace the count in the super-block by the actual count.
- NO — Ignore the error condition. This response is appropriate only if you intend to take other measures to fix the problem.

Possible responses to the SALVAGE? prompt are:

- YES — Replace the actual free-block list by a new free-block list. The new free-block list is ordered according to the gap and cylinder specifications of the **-s** or **-S** option to reduce the time spent waiting for the disk to rotate into position.
- NO — Ignore the error condition. This response is appropriate only if you intend to take other measures to fix the problem.

## Phase 5 Error Messages

### EXCESSIVE BAD BLKS IN FREE LIST (CONTINUE?)

The free-block list contains too many blocks with a value less than the first data block in the file system or greater than the last block in the file system. If you specified the **-p** option, the program terminates.

### EXCESSIVE DUP BLKS IN FREE LIST (CONTINUE?)

The free-block list contains too many blocks claimed by inodes or earlier parts of the free block list. If you specified the **-p** option, the program terminates.

### **BAD FREEBLK COUNT**

The count of free blocks in a free-list block is greater than 50 or less than 0. This condition generates the BAD FREE LIST message later in Phase 5.

### **X BAD BLKS IN FREE LIST**

X blocks in the free-block list have a block number lower than the first data block in the file system or greater than the last block in the file system. This condition generates the BAD FREE LIST message later in Phase 5.

### **X DUP BLKS IN FREE LIST**

X blocks claimed by inodes or earlier parts of the free-list block were found in the free-block list. This condition generates the BAD FREE LIST message later in Phase 5.

### **X BLK(S) MISSING**

X blocks unused by the file system were not found in the free-block list. This condition generates the BAD FREE LIST message later in Phase 5.

### **FREE BLK COUNT WRONG IN SUPERBLOCK (FIX?)**

The actual count of free blocks does not match the count of free blocks in the super-block of the file system. If you specified the **-p** option, the free block count in the super-block is fixed automatically.

### **BAD FREE LIST (SALVAGE?)**

This message is always preceded by one or more of the Phase 5 informational messages. If you specified the **-q** or **-p** option, the free-block list is salvaged automatically.

---

## **Phase 6: Salvage Free List**

This phase reconstructs the free-block list. It may display an advisory message about the blocks-to-skip or blocks-per-cylinder values.

## Phase 6 Error Messages

### DEFAULT FREE-BLOCK LIST SPACING ASSUMED

This advisory message indicates that the blocks-to-skip (gap) is greater than the blocks-per-cylinder, the blocks-to-skip is less than 1, the blocks-per-cylinder is less than 1, or the blocks-per-cylinder is greater than 500. The default values of 10 blocks-to-skip and 162 blocks-per-cylinder are used. Because the default values used may not be accurate for your system, be sure to specify correct values with the `-s` option on the command line. See the `fsck(1M)` and `mkfs(1M)` manual pages for further details.

---

## Cleanup Phase

Once it has checked a file system, `fsck` performs a few cleanup functions. The cleanup phase displays advisory messages about the file system and the status of the file system.

### Cleanup Phase Messages

#### **X files Y blocks Z free**

This advisory message indicates that the file system checked contained *X* files using *Y* blocks, and that there are *Z* blocks free in the file system.

#### **\*\*\*\*\* FILE SYSTEM WAS MODIFIED \*\*\*\*\***

This advisory message indicates that the file system was modified by `fsck`.



## Checking and Repairing ufs File Systems

This section describes the format and options of the **fsck** command for ufs file systems. For an explanation of the components it checks, see the section “ufs File System Components Checked by **fsck**”. For an explanation of phases, errors and possible responses, see the section, “Phases of **fsck** on ufs File Systems”.

---

### Format of **fsck** for s5 File Systems

The following is the ufs specific format of the **fsck** command:

```
fsck [-F ufs] [generic_options] [special...]
```

```
fsck [-F ufs] [generic_options] [-y|-Y]
[-n |-N] [-o p,b=#, w, d, r] [special...]
```

(The second command line is shown on multiple lines for readability.)

### Options

The options are as follows:

**-y** | **-Y**

Specifies a yes response for all questions. This is the normal choice when the command is being run as part of a shell procedure. **fsck** corrects all errors.

**-n** | **-N**

Specifies a no response for all questions. **fsck** does not write the file system.

- Specifies ufs file system specific suboptions. These suboptions can be any combination of the options in Figure 5–1:

| Option | Description                                                                                                                                              |
|--------|----------------------------------------------------------------------------------------------------------------------------------------------------------|
| p      | Checks the system non-interactively. Exits if there is a problem requiring intervention.                                                                 |
| b=#    | Uses the block specified as the super block for the file system. Block 32 is always an alternate super block. Other alternatives are listed during mkfs. |
| w      | Prints error message and skips to next device if device is not writable.                                                                                 |
| d      | Displays debugging messages.                                                                                                                             |
| r      | Read-only option. This option is ignored.                                                                                                                |

Figure 5–1: ufs File System Specific Options

### Sample Command Line

The following command line checks a ufs file system with no options:

```
fsck -F ufs /dev/rdisk/c0t6d0s3
```

## ufs File System Components Checked by fsck

This section discusses the kinds of consistency checks applied to each component of a ufs file system.

**fsck** checks the following components:

- Super-block
- Inodes
- Data associated with an inode
- Directory data blocks

---

### Super-Block

The most commonly corrupted item in a file system is the summary information associated with the super-block, because it is modified with every change to the blocks or inodes of the file system, and is usually corrupted after an unclean halt. The super-block is checked for inconsistencies involving:

- File system size
- Number of inodes
- Free block count
- Free inode count

### File System Size

The file system size must be larger than the number of blocks used by the super-block and the number of blocks used by the list of inodes. While there is no way to check these sizes precisely, **fsck** can check that they are within reasonable bounds. All other file system checks require that these sizes be correct. If **fsck** detects corruption in the static parameters of the default super-block, **fsck** requests the operator to specify the location of an alternate super-block.

### Free Block List

**fsck** checks that all the blocks marked as free in the cylinder group block maps are not claimed by any files. When all the blocks have been initially accounted for, **fsck** checks that the number of free blocks plus the number of blocks claimed by the inodes equals the total number of blocks in the file system. If anything is wrong with the block allocation maps, **fsck** rebuilds them, based on the list of allocated blocks it has computed.

### Free Block Count

The summary information associated with the super-block contains a count of the total number of free blocks within the file system. **fsck** compares this count to the number of free blocks it finds within the file system. If the two counts do not agree, then **fsck** replaces the incorrect count in the summary information by the actual free block count.

### Free Inode Count

The summary information contains a count of the total number of free inodes within the file system. **fsck** compares this count to the number of free inodes it finds within the file system. If the two counts do not agree, then **fsck** replaces the incorrect count in the summary information by the actual free inode count.

---

### Inodes

The list of inodes in the file system is checked sequentially starting with inode 2 (inode 0 marks unused inodes; inode 1 is saved for future generations) and progressing through the last inode in the file system. Each inode is checked for inconsistencies involving:

- Format and type
- Link count
- Duplicate blocks
- Bad block numbers
- Inode size

### Format and Type

Each inode contains a mode word. This mode word describes the type and state of the inode. Inodes may be one of six types:

- Regular
- Directory
- Symbolic link
- Special block
- Special character
- Named-pipe

Inodes may be in one of three allocation states: unallocated, allocated, and neither unallocated nor allocated. This last state means that the inode is incorrectly formatted. An inode can get into this state if bad data are written into the inode list. The only possible corrective action **fsck** can take is to clear the inode.

### Link Count

Each inode counts the total number of directory entries linked to the inode. **fsck** verifies the link count of each inode by starting at the root of the file system and descending through the directory structure. The actual link count for each inode is calculated during the descent.

Discrepancies between the link count stored in the inode and the actual link count as determined by **fsck** may be of two types:

- The stored link count is not 0 and the actual link count is 0

No directory entry appears for the inode. **fsck** places the disconnected file in the *lost+found* directory.

- The stored and actual link counts are not 0 and unequal

A directory entry may have been added or removed without the inode being updated. **fsck** replaces the incorrect stored link count by the actual link count.

Each inode contains a list, or pointers to lists (indirect blocks), of all the blocks claimed by the inode. Since indirect blocks are owned by an inode, inconsistencies in indirect blocks directly affect the inode that owns it.

### Duplicate Blocks

**fsck** compares each block number claimed by an inode against a list of already allocated blocks. If another inode already claims a block number, then the block number is added to a list of duplicate blocks. Otherwise, the list of allocated blocks is updated to include the block number.

If there are any duplicate blocks, **fsck** performs a partial second pass over the inode list to find the inode of the duplicated block. The second pass is needed, since without examining the files associated with these inodes for correct content, not enough information is available to determine which inode is corrupted and should be cleared. If this condition does arise, then the inode with the earliest modify time is usually incorrect and should be cleared. **fsck** then prompts you to clear both inodes. You must decide which one to keep and which one to clear.

### Bad Block Numbers

**fsck** checks the range of each block number claimed by an inode. If the block number is lower than the first data block in the file system, or greater than the last data block, then the block number is a bad block number. Many bad blocks in an inode are caused by an indirect block that was not written to the file system, a condition which occurs only if there has been a hardware failure. If an inode contains bad block numbers, **fsck** prompts you to clear it.

### Inode Size

Each inode contains a count of the number of data blocks that it contains. The number of actual data blocks is the sum of the allocated data blocks and the indirect blocks. **fsck** computes the actual number of data blocks and compares that block count against the actual number of blocks the inode claims. If an inode contains an incorrect count, **fsck** prompts you to fix it.

Each inode contains a thirty-two bit size field. The size is the number of data bytes in the file associated with the inode. The consistency of the byte size field is roughly checked by computing from the size field the maximum number of blocks that should be associated with the inode, and comparing that expected block count against the actual number of blocks the inode claims.

### **Data Associated with an I-Node**

An inode can directly or indirectly reference three kinds of data blocks. All referenced blocks must be the same kind. The three types of data blocks are:

- Plain data blocks
- Symbolic link data blocks
- Directory data blocks

Plain data blocks contain the information stored in a file; symbolic link data blocks contain the path name stored in a link. Directory data blocks contain directory entries. **fsck** can only check the validity of directory data blocks.

### **Directory Data Blocks**

Directory data blocks are checked for inconsistencies involving:

- Directory inode numbers pointing to unallocated inodes
- Directory inode numbers that are greater than the number of inodes in the file system
- Incorrect directory inode numbers for “.” and “..”
- Directories that are not attached to the file system

### **Directory Unallocated**

If the inode number in a directory data block references an unallocated inode, then **fsck** removes that directory entry.

### **Bad Inode Number**

If a directory entry inode number references outside the inode list, then **fsck** removes that directory entry. This condition occurs if bad data is written into a directory data block.



### Incorrect “.” and “..” Entries

The directory inode number entry for “.” must be the first entry in the directory data block. The inode number for “.” must reference itself; that is, it must equal the inode number for the directory data block. The directory inode number entry for “..” must be the second entry in the directory data block. Its value must equal the inode number for the parent of the directory entry (or the inode number of the directory data block if the directory is the root directory). If the directory inode numbers are incorrect, **fsck** replaces them with the correct values. If there are multiple hard links to a directory, the first one encountered is considered the real parent to which “..” should point; **fsck** recommends deletion for the subsequently discovered names.

### Disconnected Directories

**fsck** checks the general connectivity of the file system. If directories are not linked into the file system, then **fsck** links the directory back into the file system in the *lost+found* directory.

## Phases of fsck on ufs File Systems

---

### fsck Passes

**fsck** is a multi-pass file system check program. Each file system pass invokes a different phase of the **fsck** program. After initialization, **fsck** performs successive passes over each file system, checking blocks and sizes, path names, connectivity, reference counts, and the map of free blocks (possibly rebuilding it), and performs some cleanup.

---

### Preening File Systems

At boot time, **fsck** is normally run with the **-y** option, non-interactively. (**fsck** can also be run interactively by the administrator at any time.) **fsck** can also be run non-interactively to “preen” the file systems after an unclean halt. While preening a file system, **fsck** only fixes corruptions that are expected to result from an unclean halt. These actions are a subset of the actions that **fsck** takes when it is running interactively. When it detects an inconsistency, **fsck** generates an error message. If a response is required, **fsck** prints a prompt and waits for a response. When **fsck** is preening, most errors are fatal. For those that are expected, the response taken is noted.

This section explains the meaning of each error message, the possible responses, and the related error conditions.

### Sample Command Line

In this example, the */usr* file system is located in */dev/rdisk/clt6d0s2*. The command line below checks the */usr* file system with no options. The command operates in phases, some of which are run only if required or in response to a command line option. As each phase is completed, a message appears. The system response means that no inconsistencies were detected. At the end of the program, a summary message shows the number of files (inodes) used, blocks used, and free blocks.

```
fsck -F ufs /dev/rdisk/clt6d0s2

/dev/rdisk/clt6d0s2
File System: usr Volume: usr

** Phase 1 - Check Blocks and Sizes
** Phase 2 - Check Pathnames
** Phase 3 - Check Connectivity
** Phase 4 - Check Reference Counts
** Phase 5 - Check Free List
289 files 6522 blocks 3220 free
```

The error conditions are organized by the phase of the *fsck* program in which they can occur. The error conditions that may occur in more than one phase are discussed under initialization.

---

## Initialization Phase

Before checking a file system, **fsck** must set up certain tables and open certain files. The messages in this section relate to error conditions resulting from command line options, memory requests, the opening of files, the status of files, file system size checks, and the creation of the scratch file.

**cannot alloc *NNN* bytes for blockmap**  
**cannot alloc *NNN* bytes for freemap**  
**cannot alloc *NNN* bytes for statemap**  
**cannot alloc *NNN* bytes for Incntp**

**fsck's** request for memory for its virtual memory tables failed. This should never happen. When it does, **fsck** terminates. This is a serious system failure and should be handled immediately. Contact your service representative or another qualified person.

### **Can't open checklist file: *F***

The file system checklist or default file *F* (usually */etc/vfstab*) can not be opened for reading. When this occurs, **fsck** terminates. Check the access modes of *F*.

### **Can't stat root**

**fsck's** request for statistics about the root directory failed. This should never happen. When it does, **fsck** terminates. Contact your service representative or another qualified person.

### **Can't stat *F***

### **Can't make sense out of name *F***

**fsck's** request for statistics about the file system *F* failed. When running interactively, it ignores this file system and continues checking the next file system given. Check the access modes of *F*.

### **Can't open F**

**fsck**'s attempt to open the file system *F* failed. When running interactively, it ignores this file system and continues checking the next file system given. Check the access modes of *F*.

### **F: (NO WRITE)**

Either the **-n** flag was specified or **fsck**'s attempt to open the file system *F* for writing failed. When running interactively, **fsck** prints all the diagnostics, but does not attempt to fix anything.

### **file is not a block or character device; OK**

You have given **fsck** the name of a regular file by mistake. Check the type of the file specified.

Possible responses to the OK prompt are:

- **YES** — Ignore this error condition.
- **NO** — Ignore this file system and continue checking the next file system given.

### **UNDEFINED OPTIMIZATION IN SUPERBLOCK (SET TO DEFAULT)**

The super-block optimization parameter is neither **OPT\_TIME** nor **OPT\_SPACE**.

Possible responses to the SET TO DEFAULT prompt are:

- **YES** — Set the super-block to request optimization to minimize running time of the system. (If you want optimization to minimize disk space use, set it using **tunefs(1M)**.)
- **NO** — Ignore this error condition.

**IMPOSSIBLE MINFREE=D IN SUPERBLOCK (SET TO DEFAULT)**

The super-block minimum space percentage is greater than 99 percent or less than 0 percent.

Possible responses to the SET TO DEFAULT prompt are:

- YES — Set the MINFREE parameter to 10 percent. (If you want some other percentage, set it using `tune2fs(1M)`.)
- NO — Ignore this error condition.

**MAGIC NUMBER WRONG NCG OUT OF RANGE  
CPG OUT OF RANGE  
NCYL DOES NOT JIVE WITH NCG\*CPG  
SIZE PREPOSTEROUSLY LARGE  
TRASHED VALUES IN SUPER BLOCK**

followed by the message:

**F: BAD SUPER BLOCK: B USE -b OPTION TO FSCK TO  
SPECIFY LOCATION OF AN ALTERNATE  
SUPER-BLOCK TO SUPPLY NEEDED INFORMATION;  
SEE fsck(1M).**

The super-block has been corrupted. You must select an alternative super-block from among the available copies. Choose an alternative super-block by calculating its offset or call your service representative or another qualified person. Specifying **block 32** is a good first choice.

**INTERNAL INCONSISTENCY: M**

`fsck` has had an internal panic, whose message is *M*. This should never happen. If it does, contact your service representative or another qualified person.

### **CAN NOT SEEK: BLK B (CONTINUE)**

**fsck's** request to move to a specified block number *B* in the file system failed. This should never happen. If it does, contact your service representative or another qualified person.

Possible responses to the **CONTINUE** prompt are:

- **YES** — Attempt to continue the file system check. (Note that the problem will often persist.) This error condition prevents a complete check of the file system. Run **fsck** a second time to recheck the file system. If the block is part of the virtual memory buffer cache, **fsck** terminates with the message **Fatal I/O error**.
- **NO** — Terminate the program.

### **CAN NOT READ: BLK B (CONTINUE)**

**fsck**'s request to read a specified block number *B* in the file system failed. This should never happen. If it does, contact your service representative or another qualified person.

Possible responses to the **CONTINUE** prompt are:

- **YES** — Attempt to continue the file system check. **fsck** retries the read and prints out the message:

**THE FOLLOWING SECTORS COULD NOT BE READ: N**

where *N* indicates the sectors that could not be read. If **fsck** ever tries to write back one of the blocks on which the read failed it prints the message:

**WRITING ZERO'ED BLOCK N TO DISK**

where *N* indicates the sector that was written with zero's. If the disk is experiencing hardware problems, the problem will persist. This error condition prevents a complete check of the file system. Run **fsck** a second time to recheck the file system. If the block is part of the virtual memory buffer cache, **fsck** terminates with the message **Fatal I/O error**.

- **NO** — Terminate the program.



### **CAN NOT WRITE: BLK B (CONTINUE)**

**fsck**'s request to write a specified block number *B* in the file system failed. The disk is write-protected; check the write-protect lock on the drive. If that is not the problem, contact your service representative or another qualified person.

Possible responses to the **CONTINUE** prompt are:

- **YES** — Attempt to continue the file system check. **fsck** retries the write operation. Sectors that can not be written are indicated by the message:

THE FOLLOWING SECTORS COULD NOT BE WRITTEN: *N*

where *N* indicates the sectors that could not be written. If the disk is experiencing hardware problems, the problem will persist. This error condition prevents a complete check of the file system. Run **fsck** a second time to recheck this file system. If the block is part of the virtual memory buffer cache, **fsck** terminates with the message **Fatal I/O error**.

- **NO** — Terminate the program.

### **bad inode number *DDD* to ginode**

An internal error was caused by an attempt to read non-existent inode *DDD*. This error causes **fsck** to exit. If this occurs, contact your service representative or another qualified person.

---

## Phase 1: Check Blocks and Sizes

This phase checks the inode list. It reports error conditions encountered while:

- Checking inode types
- Setting up the zero-link-count table
- Examining inode block numbers for bad or duplicate blocks
- Checking inode size
- Checking inode format

All the errors in this phase except **INCORRECT BLOCK COUNT** and **PARTIALLY TRUNCATED INODE** are fatal if the file system is being preened.

### Phase 1 Error Messages

#### **UNKNOWN FILE TYPE I=I (CLEAR)**

The mode word of the inode *I* indicates that the inode is not a special block inode, special character inode, socket inode, regular inode, symbolic link, FIFO file, or directory inode.

Possible responses to the **CLEAR** prompt are:

- **YES** — De-allocate inode *I* by zeroing out its contents. This response always generates the **UNALLOCATED** error message in Phase 2 for each directory entry pointing to this inode.
- **NO** — Ignore this error condition.

### **PARTIALLY TRUNCATED INODE I=I (SALVAGE)**

**fsck** has found inode *I* whose size is smaller than the number of blocks allocated to it. This condition should only occur if the system crashes while truncating a file. When preening the file system, **fsck** completes the truncation to the specified size.

Possible responses to the SALVAGE prompt are:

- YES — Complete the truncation to the size specified in the inode.
- NO — Ignore this error condition.

### **LINK COUNT TABLE OVERFLOW (CONTINUE)**

An internal table for **fsck** containing allocated inodes with a link count of zero has no more room.

Possible responses to the CONTINUE prompt are:

- YES — Continue with the program. This error condition prevents a complete check of the file system. Run **fsck** a second time to recheck the file system. If another allocated inode with a zero link count is found, the error message is repeated.
- NO — Terminate the program.

### **B BAD I=I**

Inode *I* contains block number *B* with a number lower than the number of the first data block in the file system or greater than the number of the last block in the file system. This error condition may generate the EXCESSIVE BAD BLKS error message in Phase 1 if inode *I* has too many block numbers outside the file system range. This error condition generates the BAD/DUP error messages in Phases 2 and 4.

### **EXCESSIVE BAD BLKS I=I (CONTINUE)**

Too many (usually more than 10) blocks have a number lower than the number of the first data block in the file system or greater than the number of last block in the file system associated with inode *I*.

Possible responses to the CONTINUE prompt are:

- **YES** — Ignore the rest of the blocks in this inode and continue checking with the next inode in the file system. This error condition prevents a complete check of the file system. Run **fsck** a second time to recheck this file system.
- **NO** — Terminate the program.

### **BAD STATE DDD TO BLKERR**

An internal error has scrambled **fsck**'s state map to have the impossible value *DDD*. **fsck** exits immediately. If this occurs, contact your service representative or another qualified person.

### **B DUP I=I**

Inode *I* contains block number *B* that is already claimed by another inode. This error condition may generate the **EXCESSIVE DUP BLKS** error message in Phase 1 if inode *I* has too many block numbers claimed by other inodes. This error condition invokes Phase 1B and generates the **BAD/DUP** error message in Phases 2 and 4.

### **BAD MODE: MAKE IT A FILE?**

This message is generated when the status of a given inode is set to all ones, indicating file system damage. This message does not indicate disk damage, unless it appears repeatedly after **fsck -y** has been run. A response of **y** causes **fsck** to reinitialize the inode to a reasonable value.

### **EXCESSIVE DUP BLKS I=I (CONTINUE)**

Too many (usually more than 10) blocks are claimed by other inodes.

Possible responses to the CONTINUE prompt are:

- **YES** — Ignore the rest of the blocks in this inode and continue checking with the next inode in the file system. This error condition prevents a complete check of the file system. Run `fsck` a second time to recheck the file system.
- **NO** — Terminate the program.

### **DUP TABLE OVERFLOW (CONTINUE)**

An internal table in `fsck` containing duplicate block numbers has no more room.

Possible responses to the CONTINUE prompt are:

- **YES** — Continue with the program. This error condition prevents a complete check of the file system. Run `fsck` a second time to recheck the file system. If another duplicate block is found, this error message repeats.
- **NO** — Terminate the program.

### **PARTIALLY ALLOCATED INODE I=I (CLEAR)**

Inode *I* is neither allocated nor unallocated.

Possible responses to the CLEAR prompt are:

- **YES** — De-allocate inode *I* by zeroing out its contents.
- **NO** — Ignore this error condition.

**INCORRECT BLOCK COUNT I=I (X should be Y)  
(CORRECT)**

The block count for inode *I* is *X* blocks, but should be *Y* blocks. When preening, **fsck** corrects the count.

Possible responses to the **CORRECT** prompt are:

- **YES** — Replace the block count of inode *I* by *Y*.
- **NO** — Ignore this error condition.

---

**Phase 1B:  
Rescan for More  
DUPS**

When **fsck** finds a duplicate block in the file system, it rescans the file system to find the inode that previously claimed that block. When it finds the duplicate block, the following informational message appears:

**B DUP I=I**

Inode *I* contains block number *B* that is already claimed by another inode. This error condition generates the **BAD/DUP** error message in Phase 2. You can determine which inodes have overlapping blocks by examining this error condition and the **DUP** error condition in Phase 1.

---

**Phase 2: Check  
Pathnames**

This phase removes directory entries pointing to bad inodes found in Phases 1 and 1B. It reports error conditions resulting from:

- Incorrect root inode mode and status
- Directory inode pointers out of range
- Directory entries pointing to bad inodes
- Directory integrity checks

All errors in this phase are fatal if **fsck** is preening the file system, except for directories not being a multiple of the block size and extraneous hard links.

### Phase 2 Error Messages

#### ROOT INODE UNALLOCATED (ALLOCATE)

The root inode (usually inode number 2) has no allocate mode bits. This should never happen.

Possible responses to the **ALLOCATE** prompt are:

- **YES** — Allocate inode 2 as the root inode. The files and directories usually found in the root are recovered in Phase 3 and put into the *lost+found* directory. If the attempt to allocate the root fails, **fsck** exits with the message **CANNOT ALLOCATE ROOT INODE**
- **NO** — Terminate the program.

#### ROOT INODE NOT DIRECTORY (REALLOCATE)

The root inode (usually inode number 2) of the file system is not a directory inode.

Possible responses to the **REALLOCATE** prompt are:

- **YES** — Clear the existing contents of the root inode and reallocate it. The files and directories usually found in the root are recovered in Phase 3 and put into the *lost+found* directory. If the attempt to allocate the root fails, **fsck** exits with the message: **CANNOT ALLOCATE ROOT INODE**
- **NO** — **fsck** then prompts with **FIX**

Possible responses to the **FIX** prompt are:

- **YES** — Change the type of the root inode to directory. If the root inode's data blocks are not directory blocks, many error messages are generated.
- **NO** — Terminate the program.

### DUPS/BAD IN ROOT INODE (REALLOCATE)

Phase 1 or Phase 1B has found duplicate blocks or bad blocks in the root inode (usually inode number 2) of the file system.

Possible responses to the REALLOCATE prompt are:

- YES — Clear the existing contents of the root inode and reallocate it. The files and directories usually found in the root are recovered in Phase 3 and put into the *lost+found* directory. If the attempt to allocate the root fails, fsck exits with the message:

CANNOT ALLOCATE ROOT INODE

- NO — fsck then prompts with CONTINUE.

Possible responses to the CONTINUE prompt are:

- YES — Ignore the DUPS/BAD error condition in the root inode and try to continue the file system check. If the root inode is not correct, this may generate many other error messages.
- NO — Terminate the program.

### NAME TOO LONG F

An excessively long path name has been found. This usually indicates loops in the file system name space. This can occur if a privileged user has made circular links to directories. These links must be removed.

### I OUT OF RANGE I=I NAME=F (REMOVE)

A directory entry *F* has an inode number *I* that is greater than the end of the inode list.

Possible responses to the REMOVE prompt are:

- YES — Remove the directory entry *F*.
- NO — Ignore this error condition.



**UNALLOCATED I=I OWNER=O MODE=M SIZE=S  
MTIME=T TYPE=F (REMOVE)**

A directory or file entry *F* points to an unallocated inode *I*. The owner *O*, mode *M*, size *S*, modify time *T*, and name *F* are printed.

Possible responses to the REMOVE prompt are:

- YES — Remove the directory entry *F*.
- NO — Ignore this error condition.

**DUP/BAD I=I OWNER=O MODE=M SIZE=S MTIME=T  
TYPE=F (REMOVE)**

Phase 1 or Phase 1B have found duplicate blocks or bad blocks associated with directory or file entry *F*, inode *I*. The owner *O*, mode *M*, size *S*, modify time *T*, and directory name *F* are printed.

Possible responses to the REMOVE prompt are:

- YES — Remove the directory entry *F*.
- NO — Ignore this error condition.

**ZERO LENGTH DIRECTORY I=I OWNER=O MODE=M  
SIZE=S MTIME=T DIR=F (REMOVE)**

A directory entry *F* has a size *S* that is zero. The owner *O*, mode *M*, size *S*, modify time *T*, and directory name *F* are printed.

Possible responses to the REMOVE prompt are:

- YES — Remove the directory entry *F*; this generates the BAD/DUP error message in Phase 4.
- NO — Ignore this error condition.

**DIRECTORY TOO SHORT I=I OWNER=O MODE=M  
SIZE=S MTIME=T DIR=F (FIX)**

A directory *F* has been found whose size *S* is less than the minimum size directory. The owner *O*, mode *M*, size *S*, modify time *T*, and directory name *F* are printed.

Possible responses to the FIX prompt are:

- YES — Increase the size of the directory to the minimum directory size.
- NO — Ignore this directory.

**DIRECTORY F LENGTH S NOT MULTIPLE OF B  
(ADJUST)**

A directory *F* has been found with size *S* that is not a multiple of the directory block size *B*.

Possible responses to the ADJUST prompt are:

- YES — Round up the length to the appropriate block size. When preening the file system, **fsck** prints only a warning and adjusts the directory.
- NO — Ignore the error condition.

**DIRECTORY CORRUPTED I=I OWNER=O MODE=M  
SIZE=S MTIME=T DIR=F (SALVAGE)**

A directory with an inconsistent internal state has been found.

Possible responses to the SALVAGE prompt are:

- YES — Throw away all entries up to the next directory boundary (usually a 512-byte boundary). This drastic action can throw away up to 42 entries, and should be taken only after other recovery efforts have failed.
- NO — Skip to the next directory boundary and resume reading, but do not modify the directory.

**BAD INODE NUMBER FOR '.' I=I OWNER=O MODE=M  
SIZE=S MTIME=T DIR=F (FIX)**

A directory *I* has been found whose inode number for '.' does not equal *I*.

Possible responses to the FIX prompt are:

- YES — Change the inode number for '.' to be equal to *I*.
- NO — Leave the inode number for '.' unchanged.

**MISSING '.' I=I OWNER=O MODE=M SIZE=S MTIME=T  
DIR=F (FIX)**

A directory *I* has been found whose first entry is unallocated.

Possible responses to the FIX prompt are:

- YES — Build an entry for '.' with inode number equal to *I*.
- NO — Leave the directory unchanged.

**MISSING '.' I=I OWNER=O MODE=M SIZE=S MTIME=T  
DIR=F CANNOT FIX, FIRST ENTRY IN DIRECTORY  
CONTAINS F**

A directory *I* has been found whose first entry is *F*. *fsck* cannot resolve this problem. Mount the file system and move entry *F* elsewhere. Then unmount the file system and run *fsck* again.

**MISSING '.' I=I OWNER=O MODE=M SIZE=S MTIME=T  
DIR=F CANNOT FIX, INSUFFICIENT SPACE TO ADD '.'**

A directory *I* has been found whose first entry is not '.'. This should never happen. *fsck* cannot resolve the problem. If this occurs, contact your service representative or another qualified person.

**EXTRA ‘.’ ENTRY I=I OWNER=O MODE=M SIZE=S  
MTIME=T DIR=F (FIX)**

A directory *I* has been found that has more than one entry for ‘.’.

Possible responses to the FIX prompt are:

- YES — Remove the extra entry for ‘.’.
- NO — Leave the directory unchanged.

**BAD INODE NUMBER FOR ‘.’ I=I OWNER=O MODE=M  
SIZE=S MTIME=T DIR=F (FIX)**

A directory *I* has been found whose inode number for ‘.’ does not equal the parent of *I*.

Possible responses to the FIX prompt are:

- YES — Change the inode number for ‘.’ to be equal to the parent of *I*. (Note that ‘.’ in the root inode points to itself.)
- NO — Leave the inode number for ‘.’ unchanged.

**MISSING ‘.’ I=I OWNER=O MODE=M SIZE=S MTIME=T  
DIR=F (FIX)**

A directory *I* has been found whose second entry is unallocated.

Possible responses to the FIX prompt are:

- YES — Build an entry for ‘.’ with inode number equal to the parent of *I*. (Note that ‘.’ in the root inode points to itself.)
- NO — Leave the directory unchanged.

**MISSING '..' I=I OWNER=O MODE=M SIZE=S MTIME=T  
DIR=F CANNOT FIX, SECOND ENTRY IN DIRECTORY  
CONTAINS F**

A directory *I* has been found whose second entry is *F*. **fsck** cannot resolve this problem. Mount the file system and move entry *F* elsewhere. Then remount the file system and run **fsck** again.

**MISSING '..' I=I OWNER=O MODE=M SIZE=S MTIME=T  
DIR=F CANNOT FIX, INSUFFICIENT SPACE TO ADD '..'**

A directory *I* has been found whose second entry is not '..' (the parent directory). **fsck** cannot resolve this problem. Mount the file system and move the second entry in the directory elsewhere. Then unmount the file system and run **fsck** again.

**EXTRA '..' ENTRY I=I OWNER=O MODE=M SIZE=S  
MTIME=T DIR=F (FIX)**

A directory *I* has been found that has more than one entry for '..' (the parent directory).

Possible responses to the **FIX** prompt are:

- YES — Remove the extra entry for '..' (the parent directory).
- NO — Leave the directory unchanged.

**N IS AN EXTRANEIOUS HARD LINK TO A DIRECTORY  
D (REMOVE)**

**fsck** has found a hard link *N* to a directory *D*. When preening, **fsck** ignores the extraneous links.

Possible responses to the **REMOVE** prompt are:

- YES— Delete the extraneous entry *N*.
- NO — Ignore the error condition.

### **BAD INODE S TO DESCEND**

An internal error has caused an impossible state *S* to be passed to the routine that descends the file system directory structure. **fsck** exits. If this occurs, contact your service representative or another qualified person.

### **BAD RETURN STATE S FROM DESCEND**

An internal error has caused an impossible state *S* to be returned from the routine that descends the file system directory structure. **fsck** exits. If you encounter this error, contact your service representative or another qualified person.

### **BAD STATE S FOR ROOT INODE**

An internal error has caused an impossible state *S* to be assigned to the root inode. **fsck** exits. If this occurs, contact your service representative or another qualified person.

---

## **Phase 3: Check Connectivity**

This phase checks the directories examined in Phase 2. It reports error conditions resulting from:

- Unreferenced directories
- Missing or full *lost+found* directories

### **Phase 3 Error Messages**

**UNREF DIR I=I OWNER=O MODE=M SIZE=S MTIME=T (RECONNECT)**

The directory inode *I* was not connected to a directory entry when the file system was traversed. The owner *O*, mode *M*, size *S*, and modify time *T* of directory inode *I* are printed. When preening, **fsck** reconnects the director if its size is non-zero; otherwise it clears the directory.

Possible responses to the RECONNECT prompt are:

- **YES** — Reconnect directory inode *I* to the file system in the directory for lost files (usually the *lost+found* directory). This may generate the *lost+found* error messages in Phase 3 if there are problems connecting directory inode *I* to the *lost+found* directory. It may also generate the **CONNECTED** error message in Phase 3 if the link was successful.
- **NO** — Ignore this error condition. This response generates the **UNREF** error message in Phase 4.

### **NO *lost+found* DIRECTORY (CREATE)**

There is no *lost+found* directory in the root directory of the file system. When preening, **fsck** tries to create a *lost+found* directory.

Possible responses to the **CREATE** prompt are:

- **YES** — Create a *lost+found* directory in the root of the file system. This may produce the message:

NO SPACE LEFT IN / (EXPAND)

Inability to create a *lost+found* directory generates the message:

**SORRY. CANNOT CREATE *lost+found* DIRECTORY**

and aborts the attempt to link up the lost inode. This in turn generates the **UNREF** error message in Phase 4.

- **NO** — Abort the attempt to link up the lost inode. This generates the **UNREF** error message in Phase 4.

### **lost+found IS NOT A DIRECTORY (REALLOCATE)**

The entry for *lost+found* is not a directory.

Possible responses to the REALLOCATE prompt are:

- YES — Allocate a directory inode, and change *lost+found* to reference it. The previous inode referenced by the *lost+found* directory is not cleared. Thus it is either reclaimed as an UNREF'ed inode or has its link count ADJUST'ed later in this phase.

Inability to create a *lost+found* directory generates the message:

**SORRY. CANNOT CREATE *lost+found* DIRECTORY**

and aborts the attempt to link up the lost inode. This in turn generates the UNREF error message in Phase 4.

- NO — Abort the attempt to link up the lost inode. This response generates the UNREF error message in Phase 4.

### **NO SPACE LEFT IN /lost+found (EXPAND)**

There is no space to add another entry to the *lost+found* directory in the root directory of the file system. When preening, fsck expands the *lost+found* directory.

Possible responses to the EXPAND prompt are:

- YES — Expand the *lost+found* directory to make room for the new entry. If the attempted expansion fails, fsck prints the message:

**SORRY. NO SPACE IN *lost+found* DIRECTORY**

and aborts the attempt to link up the lost inode. This in turn generates the UNREF error message in Phase 4. Clear out unnecessary entries in the *lost+found* directory. This error is fatal if the file system is being preened.



- NO — Abort the attempt to link up the lost inode. This response generates the UNREF error message in Phase 4.

### **DIR I=I1 CONNECTED. PARENT WAS I=I2**

This advisory message indicates that a directory inode *I1* was successfully connected to the *lost+found* directory. The parent inode *I2* of the directory inode *I1* is replaced by the inode number of the *lost+found* directory.

### **DIRECTORY F LENGTH S NOT MULTIPLE OF B (ADJUST)**

A directory *F* has been found with size *S* that is not a multiple of the directory block size *B*. (Note that this may reoccur in Phase 3 if the error condition is not corrected in Phase 2).

Possible responses to the ADJUST prompt are:

- YES — Round up the length to the appropriate block size. When preening the file system, **fsck** prints only a warning and adjusts the directory *F*.
- NO — Ignore the error condition.

### **BAD INODE S TO DESCEND**

An internal error has caused an impossible state *S* to be passed to the routine that descends the file system directory structure. **fsck** exits. If this occurs, contact your service representative or another qualified person.

---

## **Phase 4: Check Reference Counts**

This phase checks the link count information obtained in Phases 2 and 3. It reports error conditions resulting from:

- Unreferenced files
- Missing or full *lost+found* directory
- Incorrect link counts for files, directories, symbolic links, or special files

- Unreferenced files, symbolic links, and directories
- Bad or duplicate blocks in files, symbolic links, and directories

All errors in this phase (except running out of space in the *lost+found* directory) are correctable if the file system is being preened.

### Phase 4 Error Messages

**UNREF FILE I=I OWNER=O MODE=M SIZE=S  
MTIME=T (RECONNECT)**

Inode *I* was not connected to a directory entry when the file system was traversed. The owner *O*, mode *M*, size *S*, and modify time *T* of inode *I* are printed. When preening, fsck clears the file if either its size or its link count is zero; otherwise fsck reconnects it.

Possible responses to the RECONNECT prompt are:

- YES — Reconnect inode *I* to the file system in the directory for lost files (usually the *lost+found* directory). This may generate the *lost+found* error message in Phase 4 if there are problems connecting inode *I* to the *lost+found* directory.
- NO — Ignore this error condition. This response always invokes the CLEAR error condition in Phase 4.

### (CLEAR)

The inode mentioned in the error message immediately preceding can not be reconnected. This message cannot appear if fsck is preening the file system, because lack of space to reconnect files is a fatal error.

Possible responses to the CLEAR prompt are:

- YES — De-allocate the inode by zeroing out its contents.
- NO — Ignore this error condition.

### **NO *lost+found* DIRECTORY (CREATE)**

There is no *lost+found* directory in the root directory of the file system. When precning, **fsck** tries to create a *lost+found* directory.

Possible responses to the CREATE prompt are:

- **YES** — Create a *lost+found* directory in the root of the file system. This may generate the message:

NO SPACE LEFT IN / (EXPAND)

Inability to create a *lost+found* directory generates the message:

SORRY. CANNOT CREATE *lost+found* DIRECTORY

and aborts the attempt to link up the lost inode. This in turn generates UNREF error message in Phase 4.

- **NO** — Abort the attempt to link up the lost inode. This response generates the UNREF error message in Phase 4.

**lost+found IS NOT A DIRECTORY (REALLOCATE)**

The entry for *lost+found* is not a directory.

Possible responses to the REALLOCATE prompt are:

- YES — Allocate a directory inode and change the *lost+found* directory to reference it. The previous inode reference by the *lost+found* directory is not cleared. Thus it is either reclaimed as an UNREF'ed inode or has its link count ADJUST'ed later in this phase. Inability to create a *lost+found* directory generates the message:

SORRY. CANNOT CREATE *lost+found* DIRECTORY

and aborts the attempt to link up the lost inode. This generates the UNREF error message in Phase 4.

- NO — Abort the attempt to link up the lost inode. This response generates the UNREF error message in Phase 4.

### **NO SPACE LEFT IN /lost+found (EXPAND)**

There is no space to add another entry to the *lost+found* directory in the root directory of the file system. When preening, **fsck** expands the *lost+found* directory.

Possible responses to the EXPAND prompt are:

- **YES** — Expand the *lost+found* directory to make room for the new entry. If the attempted expansion fails, **fsck** prints the message

**SORRY. NO SPACE IN lost+found DIRECTORY**

and aborts the attempt to link up the lost inode. This generates the UNREF error message in Phase 4. Clear out unnecessary entries in the *lost+found* directory. This error is fatal if the file system is being preened.

- **NO** — Abort the attempt to link up the lost inode. This response generates the UNREF error message in Phase 4.

### **LINK COUNT TYPE I=I OWNER=O MODE=M SIZE=S MTIME=T COUNT=X SHOULD BE Y (ADJUST)**

The link count for inode *I* is *X* but should be *Y*. The owner *O*, mode *M*, size *S*, and modify time *T* are printed. When preening, **fsck** adjusts the link count unless the number of references is increasing, a condition that should never occur unless precipitated by a hardware failure. When the number of references is increasing during preening, **fsck** exits with the message:

**LINK COUNT INCREASING**

Possible responses to the ADJUST prompt are:

- **YES** — Replace the link count of file inode *I* by *Y*.
- **NO** — Ignore this error condition.

**UNREF TYPE I=I OWNER=O MODE=M SIZE=S  
MTIME=T (CLEAR)**

Inode *I* was not connected to a directory entry when the file system was traversed. The owner *O*, mode *M*, size *S*, and modify time *T* of inode *I* are printed. Since this file was not connected because its size or link count was zero, **fsck** clears it during preening.

Possible responses to the **CLEAR** prompt are:

- **YES** — De-allocate inode *I* by zeroing out its contents.
- **NO** — Ignore this error condition.

**BAD/DUP TYPE I=I OWNER=O MODE=M SIZE=S  
MTIME=T (CLEAR)**

Phase 1 or Phase 1B has found duplicate blocks or bad blocks associated with inode *I*. The owner *O*, mode *M*, size *S*, and modify time *T* of inode *I* are printed. This message cannot appear when the file system is being preened, because it would have caused a fatal error earlier.

Possible responses to the **CLEAR** prompt are:

- **YES** — De-allocate inode *I* by zeroing out its contents.
- **NO** — Ignore this error condition.

---

### Phase 5: Check Cylinder Groups

This phase checks the free block and used inode maps. It reports error conditions resulting from:

- Allocated blocks in the free block maps
- Free blocks missing from free block maps
- Incorrect total free block count
- Free inodes in the used inode maps
- Allocated inodes missing from used inode maps
- Incorrect total used inode count

### Phase 5 Error Messages

#### CG C: BAD MAGIC NUMBER

The magic number of cylinder group *C* is wrong. This message usually indicates that the cylinder group maps have been destroyed. When running interactively, the cylinder group is marked as needing reconstruction. This error is fatal if the file system is being preened.

#### BLK(S) MISSING IN BIT MAPS (SALVAGE)

A cylinder group block map is missing some free blocks. During preening, fsck reconstructs the maps.

Possible responses to the SALVAGE prompt are:

- YES — Reconstruct the free block map.
- NO — Ignore this error condition.

### **SUMMARY INFORMATION BAD (SALVAGE)**

The summary information was found to be incorrect. When preening, **fsck** recomputes the summary information.

Possible responses to the SALVAGE prompt are:

- YES — Reconstruct the summary information.
- NO — Ignore this error condition.

### **FREE BLK COUNT(S) WRONG IN SUPERBLOCK**

(SALVAGE) The super-block free block information was found to be incorrect. When preening, **fsck** recomputes the super-block free block information.

Possible responses to the SALVAGE prompt are:

- YES — Reconstruct the super-block free block information.
- NO — Ignore this error condition.

---

## **Cleanup Phase**

After checking a file system, **fsck** performs a few cleanup functions. The cleanup phase displays advisory messages about the status of the file system.

### **V files, W used, X free (Y frags, Z blocks)**

This advisory message indicates that the file system checked contains *V* files using *W* fragment sized blocks, and that there are *X* fragment sized blocks free in the file system. The numbers in parentheses break the free count down into *Y* free fragments and *Z* free full sized blocks.

### **\*\*\*\*\* FILE SYSTEM WAS MODIFIED \*\*\*\*\***

This advisory message indicates that the file system was modified by **fsck**. If this file system is mounted or is the current root file system, you should reboot. If the file system is mounted, you may need to unmount it and run **fsck** again; otherwise the work done by **fsck** may be undone by the in-core copies of tables.



## Checking bfs File Systems

All I/O for bfs file systems is synchronous; therefore, bfs file systems do not get corrupted even if proper shutdown procedures are not observed.

Corruption of a bfs file system is likely to occur only if the system crashes during the process of compaction. Compaction is described in Appendix A.

**fsck** checks the sanity words stored in the bfs super-block to see if compaction was in process before the system crashed. If it was, **fsck** completes the compaction of the file system.

This chapter contains procedures for backing up information on your system.

|                                                   |      |
|---------------------------------------------------|------|
| What You Should Know Before You Perform a Backup  | 6-2  |
| Backing Up File Systems .....                     | 6-4  |
| Backing Up Individual Files and Directories ..... | 6-10 |
| Backing Up Raw Devices .....                      | 6-14 |
| Backing Up File System and Disk Information ..... | 6-16 |
| Backing Up the Entire System .....                | 6-19 |
| Backing Up an Entire NCR 3600 .....               | 6-20 |
| Copying Files to NCR TOWERS .....                 | 6-25 |
| Retensioning a Cartridge Tape .....               | 6-26 |

## What You Should Know Before You Perform a Backup

- You can also perform backups and restores using the **sysadm** or **OSA** (Open System Administrator) menus. However, each method of backing up and restoring offers different features. Refer to the *NCR UNIX SVR4 Administrator Guide: OA&M Menu Interface* and *Open Systems Administrator Guide* for more information.
- When you use command line procedures that include SCSI device names, be sure to substitute the appropriate device names for your particular system. Refer to Appendix B for more information.
- If you are using Volume Manager, you must have Volume Manager running on your system to restore file systems under Volume Manager control to the correct locations. If Volume Manager is not running, the file systems can not be mounted. Therefore, file systems under Volume Manager control are restored under the *root* file system, not as separate file systems.
- The */usr* file system is very sensitive; restoring files in */usr* in multiuser mode can result in file system corruption. In addition, most of the utilities and libraries used to restore data reside under */usr*, so overwriting them could destroy the file, prohibiting further restores. Do NOT restore the */usr* file system in multiuser mode.
- The file systems */dev/fd* and */dev/proc* should never be restored.

- If the *root* or */usr* file system is a vxfs file system or is under Volume Manager control, it can NOT be restored through maintenance. OSA provides the capability for restoring these types of file systems. Therefore, you should use OSA to back it up and restore it.
- If */usr* is a separate file system, you must make sure that the user *root* is using the Bourne shell, and NOT the Korn shell upon login. You can define the default shell to be used upon login in the last field of the root entry in the */etc/passwd* file. This will avoid the problem of not being able to log into your system after reboot upon restoring the *root* and */stand* file system from the maintenance mode (you will be kept in a loop where the system prompt you for either the root password or a new run level). Because the Korn shell is located under the */usr/bin* directory, the root user will not be able to login and will not be able to bring the system into single user mode for further restores in the case where */usr* is missing as a separate file system.

## Backing Up File Systems

---

### Why/When

You can use this procedure to back up the complete system or a single file system. You should always have a current backup of each file system on your system.

**NOTE:** To recover a complete system or disk, you must also have a backup of file system and disk information. See “Backing Up File System and Disk Information” in this chapter.

---

### Before You Start

To prevent files from being modified during the backup, put the system in single user mode (run level *s*) or be sure no one is using the data you are about to back up. You may want to unmount any file systems involved and mount them again as read-only.

**NOTE:** Run level 1 does NOT prevent users from using the system.

Before you back up or copy files from a file system, you should always unmount the file system and perform the **fsck** utility on that file system. For the *root* file system, perform a shutdown and reboot (**shutdown -i6 -g0 -y**) before performing the backup.

You must create a backup list for every slice which contains a file system. For example, do not create a backup list containing the files from both the *root* file system (slice1) and the */stand* file system (slice a). This is recommended so you avoid restoring more than one file system to the same physical slice. Backing up a file system involves the following:

- Creation of a backup list containing the files to be backed up
- Backing up the file system data from the created backup list

## Creating Backup Lists

Backup lists for individual file systems can be created by using the command **find** with the option **-mount**. This option limits the search to the file system containing the specified directory. For example, the command **find /home -mount -print** will only list files in the */home* file system ignoring any file from file systems mounted under */home*.

If you only want to backup the operating system, not including any user-defined file systems such as */home*, you must perform the procedures “Creating a Backup List for the Root File System”, “Creating a Backup List for the */stand* File System” and if */usr*, */var* or */tmp* are on separate slices you must create a backup list for each of those as well following the appropriate procedures below.

## Creating a Backup List for the Root File System

To restore the root file system separately, you must backup the root file system separately from any other file systems. To backup the root file system on slice 1, perform the following steps:

Step 1: Login as **root** and create a backup list containing the files in the *root* slice only. Use the following commands:

```
cd /
find . -mount -print >/tmp/backup.root
```

Step 2: Add the directory entries */proc*, */dev/fd*, and any other mounted file system such as */usr* or */var* to the backup list. Although any files in the special file systems of */proc* and */dev/fd* cannot be backed up, their root directories must be added to the backup list. At restore time, these special file systems can only be re-created properly if their directory mount point already exists.

```
echo "/proc" >>/tmp/backup.root
echo "/dev/fd" >>/tmp/backup.root
```

---

### Creating a Backup List for the */stand* File System

Since the */stand* file system is always on a separate slice, you should create a separate backup list, performing the following step:

- Step 1: Login as **root** and create a backup list containing the files in the *stand* slice only. Use the following commands:

```
cd /stand
find . -print >/tmp/backup.stand
```

**NOTE:** You can omit the option **-mount** in the **find** command because no file systems should be mounted under */stand*.

---

### Creating a Backup List for the */usr* File System

Create a separate backup list for the */usr* directory only if */usr* is on a separate slice.

- Step 1: Login as **root** and create a backup list containing the files in the */usr* slice only. Use the following commands:

```
cd /usr
find . -print >/tmp/backup.usr
```

**NOTE:** You can omit the option **-mount** in the **find** command because no file systems should be mounted under */usr*.

---

### Creating a Backup List for the */var* File System

Create a separate backup list for the */var* directory only if */var* is on a separate slice.

- Step 1: Login as **root** and create a backup list containing the files in the */var* slice only. Use the following commands:

```
cd /var
find . -mount -print >/tmp/backup.var
```

## Creating a Backup List for the */tmp* File System

Create a separate backup list for the */tmp* directory only if */tmp* is on a separate slice.

Step 1: Before backing up the */tmp* file system, delete any unnecessary files. Since the */tmp* directory should contain only temporary data and since the contents under */tmp* will be deleted at system startup time, you should only have to backup this directory as a file system if there are specific user created files that need to be saved.

Step 2: Login as **root** and create a backup list containing the files in the */tmp* slice only. Use the following commands:

```
cd /tmp
find . -mount -print >/tmp/backup.tmp
```

## Creating a Backup List for the */var/tmp* File System

Create a separate backup list for the */var/tmp* directory only if */var/tmp* is on a separate slice.

Step 1: Before backing up the */var/tmp* file system, delete any unnecessary files. Since the */var/tmp* directory should contain only temporary data and since the contents under */var/tmp* will be deleted at system startup time, you should only have to backup this directory as a file system if there are specific user created files that need to be saved.

Step 2: Login as **root** and create a backup list containing the files in the */var/tmp* slice only. Use the following commands:

```
cd /var/tmp
find . -mount -print >/tmp/backup.vartmp
```



---

### Creating a Backup List for a User-Defined File Systems

To restore a user-defined file system separately or to restore an entire disk or system you must backup each user-defined file system (for example, */home*) separately.

Step 1: Login as **root**, change to the mount point directory of the file system you are backing up and create a file that contains the names of the files you want to back up. For example:

```
cd mount-point
find . -mount -print >/tmp/backup.mount_point
```

---

### Creating a Backup List for User-Defined File Systems

To restore a user-defined file system separately or to restore an entire disk or system, you must back up each user-defined file system (for example, */home*) separately.

Log in as **root**, change to the mount point directory of the file system you are backing up, and create a file that contains the names of the files you want to back up. For example:

```
cd mount_point
find . -mount -print > /tmp/backup.mount_point
```

---

### Backing up from Backup Lists

You can choose different backup medias for your backups. For example, you can backup to cartridge tape, to a helical tape or to flex diskette. See Appendix B, "Device Names and Numbers," for an explanation of device names for backup devices.

If you are using flex diskettes, consider the following:

- Only backup small amounts of data
- To format a double sided, high density diskette of 1.44 MB (2880 blocks @ 512 bytes), enter the command:

```
format /dev/rdisk/f03ht
```

- Once a flex diskette is formatted, refer to it as `/dev/rdsk/f0t`. The device name `/dev/rdsk/f0t` does not specify the density of the device. Once the flex diskette is formatted, and you access the device with utilities such as `cpio`, the device driver automatically determines whether the flex diskette is high density.

Step 1: Change to the mount directory of the file system you want to back up:

```
cd mount_point
```

*mount\_point* is the directory containing the files in the backup list.

Step 2: Insert a backup medium in the appropriate drive and type the following command:

```
cpio -ocvB < /tmp/backup_list \
-O /dev/rmt/device_name
```

The `-O` option permits multiple-volume backups. If, for example, you are using cartridge tapes for backups and a cartridge tape becomes full, the following messages are displayed.

```
End of medium on "output".
Change to part X and press RETURN key. [q]
```

The letter *X* indicates the sequence (part) number of the backup media. For example, the first backup media should be labeled **part 1**, the second backup media that becomes full should be labeled **part 2**, etc.

Step 3: If you are backing up to multiple volumes, continue the backup by inserting the next backup media and pressing the **ENTER** key.

To terminate the backup, enter the letter **q** (quit).

## Backing Up Individual Files and Directories

---

### Before You Start

To prevent files from being modified during the backup, put the system in single user mode (run level *s*) or be sure no one is using the data you are about to back up. You may want to unmount any file systems involved and mount them again as read-only.

**NOTE:** Run level 1 does NOT prevent users from using the system.

If you are using flex diskettes, consider the following:

- Only backup small amounts of data
- To format a double sided, high density diskette of 1.44 MB (2880 blocks @ 512 bytes), enter the command:  
  

```
format /dev/rdisk/f03ht
```
- Once a flex diskette is formatted, refer to it as */dev/rdisk/f0t*. The device name */dev/rdisk/f0t* does not specify the density of the device. Once the flex diskette is formatted, and you access the device with utilities such as **cpio**, the device driver automatically determines whether the flex diskette is high density.

---

## Backing Up a Single Directory

Perform the following steps to backup a single directory:

Step 1: Log in as **root**, change to the directory you are backing up and create the file list. For example:

```
cd directory
find . -print | cpio -ocvB > \
/dev/rmt/tape_device
```

**NOTE:** If you create a backup containing relative pathnames (with the **find .** command), you will be able to restore the directory to a different location. However if you want the directory restored to the same original location, you must **cd** to that original directory before restoring its contents. If you want to always be sure you are restoring a directory to its original location, you can use the following alternate command to backup:

```
find absolute_pathname_of_directory -print |
cpio -ocvB > /dev/rmt/device_name
```

### Backing Up Multiple Directories

Perform the following steps to backup up multiple directories:

Step 1: Login as **root** and type in the following command:

```
find list_of_absolute_pathnames -print | cpio
-ocvB -O /dev/rmt/device_name
```

The *list\_of\_absolute\_pathnames* consists of list of absolute pathnames. For example, to backup the directories */home/user1*, */home/user2*, and */home/user3* using the tape device *c0t3d0s0*, type the following command:

```
find /home/user1 /home/user2 /home/user3 -print |
cpio -ocvB -O /dev/rmt/c0t3d0s0
```

The **-O** option permits multiple-volume backups. If, for example, you are using cartridge tapes for backups and a cartridge tape becomes full, the following messages are displayed.

```
End of medium on "output".
Change to part X and press RETURN key. [q]
```

The letter *X* indicates the sequence (part) number of the backup media. For example, the first backup media should be labeled **part 1**, the second backup media that becomes full should be labeled **part 2**, etc.

Step 2: If you are backing up to multiple volumes, continue the backup by inserting the next backup media and pressing the **ENTER** key.

To terminate the backup, enter the letter **q** (quit).

## Backing Up Individual Files

Perform the following steps to backup individual files:

Step 1: Login as root and type in the following command:

```
find absolute_pathnames -print | cpio -ocvB -O /dev/rmt/device_name
```

Examples are:

```
find /home/user1/*.c -print | cpio -ocvB -O /dev/rmt/c0t3d0s0
```

```
find /home/user?/prog[1-5].c | cpio -ocvB -O /dev/rmt/c0t3d0s0
```

```
find / -name "*.dbf" -print | cpio -ocvB -O /dev/rmt/c0t3d0s0
```

The `-O` option permits multiple-volume backups. If, for example, you are using cartridge tapes for backups and a cartridge tape becomes full, the following messages are displayed.

```
End of medium on "output".
Change to part X and press RETURN key. [q]
```

The letter `X` indicates the sequence (part) number of the backup media. For example, the first backup media should be labeled **part 1**, the second backup media that becomes full should be labeled **part 2**, etc.

Step 2: If you are backing up to multiple volumes, continue the backup by inserting the next backup media and pressing the **ENTER** key.

To terminate the backup, enter the letter **q** (quit).

## Backing Up Raw Devices

Use this procedure to backup raw devices. A raw device is a slice on a disk without a file system. For example, if you have a database installed on raw slices, you can backup each raw slice individually.

---

### Before You Start

To prevent files from being modified during the backup, put the system in single user mode (run level `s`) or be sure no one is using the data you are about to back up.

For databases residing on raw slices, you can shutdown the database to make the data on the raw slices inaccessible to users.

Make sure no daemons are running which could write to the raw data slices. The following command displays the process IDs of the processes using the device:

```
fuser /dev/dsk/device_name
```

Backups of raw devices are done using the `dd` command. This means that the backup is a physical byte-by-byte backup of physical blocks and not a logical file-by-file backup as with for example the `cpio` command. Therefore, you cannot backup individual files or directories residing in raw slices with this method.

---

## Procedure

For each raw slice, perform the following step:

Step 1: To backup a raw data slice, use the following command:

```
dd if=/dev/rdisk/slice_name bs=512k
of=/dev/rmt/device_name
```

For example, to backup the slice `/dev/rdisk/c0t0d0s4` to cartridge tape `/dev/rmt/c0t3d0s0`, type the following command:

```
dd if=/dev/rdisk/c0t0d0s4 bs=512k
of=/dev/rmt/c0t3d0s0
```



## Backing Up File System and Disk Information

To recover a corrupt or destroyed root disk without re-installing, and to restore any non-root disk to its original configuration you must have the necessary backups for each file system as well as backups of the following:

- A copy of your */etc/vfstab* file
- A copy of the *vtoc* (volume table of contents) structure that contains UNIX slicing information for each disk
- Partitioning information for each disk
- Parameters used to create the file systems

If your maintenance diskette does not have enough room left, you can create an *s5* file system on another diskette and then copy the information to that diskette. When you need to retrieve the information, simply mount that additional flex diskette. For a mount point, create a directory (for example, */mnt2*). It is a good idea to have a print out of the disk layout information.

**NOTE:** Make sure you have a backup copy of your original maintenance diskette. If you do not have one, create a backup copy using the procedure “Copying the Boot and Maintenance Flex Diskettes” in Chapter 2.

## Procedure

Use the following procedure to create the necessary backups on your maintenance diskette:

Step 1: Insert the maintenance flex diskette in flex drive 0.

Step 2: Mount the file system, make a restore directory and change directory to it:

```
mount -F s5 /dev/dsk/f0 /mnt
mkdir /mnt/restore
cd /mnt/restore
```

**NOTE:** Make sure the maintenance diskette is write-enabled. If the diskette is read-only protected, the **mount** command returns the error message "no such device."

Step 3: Enter the following command to copy the */etc/vfstab* file to the maintenance diskette:

```
cp /etc/vfstab /mnt/restore
```

Step 4: For each disk, perform this step to save the **vtoc** information. Use a different file name for each disk. For example, to save the information for the disk *c0t6d0s0* to a file named *c0t6d0s0*, enter the following:

```
prtvtoc -f c0t6d0s0 /dev/rdsk/c0t6d0s0
```

Step 5: For each disk, perform the **fdisk** command to obtain partitioning information for the disk. For example, to save the partition information for the disk *c0t6d0s0* to a file named *fdisk6*, enter the following:

```
fdisk -l /dev/rdsk/c0t6d0s0 > fdisk6
```

Step 6: For each file system save the parameters that were used when creating the file system. For example, if you have */home* installed as a separate file system and it resides on */dev/dsk/c0t5d0s1* as a *ufs* type file system, to save the information, enter:

```
mkfs -F ufs -m /dev/dsk/c0t5d0s1 > fs.home
```

## Backing Up Information

**Step 7:** Print a copy of all the files you just created under */mnt/restore* and keep these hard copies with the backup tapes.

**Step 8:** Unmount the maintenance file system by entering the following commands:

```
cd /
umount /mnt
```

**Step 9:** If you do not have recent backups of your file systems, you should back them up at this time. Refer to the procedure “Backing up File Systems” in this chapter.

## Backing Up the Entire System

You can use the following procedure to backup the entire system from the command line.

---

### Procedure

Perform the following steps to backup the entire system:

- Step 1: Follow the procedure “Backing up File System and Disk Information” in this chapter.
- Step 2: Follow the procedure “Backing up File Systems” in this chapter.
- Step 3: Follow the procedure “Backing up Raw Devices”.

## Backing Up an Entire NCR 3600

Use the following procedure to perform a full system backup of an NCR 3600 AP. Note that this procedure uses both the command line interface and the **sysadm** menus. It backs up all files except those in */proc*, */dev/fd*, and */tmp*, and any NFS mounted file systems.

This procedure consists of the following tasks:

- Configuring the backup
- Verifying tape device functionality
- Performing the backup

---

### Configuring the Backup

Perform the following steps to configure your NCR 3600 AP for a full system backup.

**NOTE:** You need only perform this procedure once after installing or upgrading your NCR 3600 AP. If you are unsure whether or not the steps have been performed, check the contents of */etc/Backup* and */etc/Ignore* and compare them to the contents listed in Steps 2 and 3.

Step 1: Save the original */etc/Backup* and */etc/Ignore* files that were provided with the base installed operating system:

```
mv /etc/Backup /etc/Backup.orig
mv /etc/Ignore /etc/Ignore.orig
```

Step 2: Create a new */etc/Backup* file that contains the following entry:

```
/
```

Step 3: Create a new */etc/ignore* file that contains the following entries:

```
/proc
/dev/fd
/tmp
```

Also include any NFS mounted file system in */etc/ignore*. To identify NFS mounted file systems, enter the following command:

```
mount
```

The following is an example of the **mount(1M)** display for an NFS file system mounted on */mnt*. In this case, you would add */mnt* to */etc/ignore*.

```
/mnt on Thames: /export/d6 on Fri Apr 19
10:41:55 1992
.
```

## Verifying Tape Device Functionality

Perform the following steps to verify the functionality of the tape device on an NCR 3600 AP.

Step 1: Place a tape in the tape device. All data currently on the tape will be destroyed.

Step 2: Change to the */tmp* directory:

```
cd /tmp
```

Step 3: Enter the following command to write to the tape:

```
find . -print | cpio -ocv > /dev/rmt/device
```

where *device* is the device name of your tape device.

Step 4: When the tape stops, enter the following command to read the information you just wrote to tape:

```
cpio -icvt < /dev/rmt/device
```

## Backing Up Information

The **cpio(1)** command displays a table of contents for the tape which should look like the list printed when you wrote the tape.

If you receive any errors, use the **devstat(1)** command to display the device status:

```
devstat -d /dev/rmt/device
```

The state should be as follows:

```
STATE: ----
```

If the state is not clear, another process has the device. Before you continue, you must free the device. Likely suspects would be ASF running or a process that died while it has ownership of the tape unit.

---

## Performing the Backup

Perform the following steps to write a backup copy of your system to tape. This task requires the AP to be in single user mode and takes several hours to complete. You are not required to monitor the backup after it starts; however, you must ensure that the AP is left alone until you have completed the backup.

Step 1: Open a console window from the AWS to the AP that the backup is to be performed on and set the terminal type as follows (Korn shell and Bourne shell only):

```
TERM=motifterm; export TERM
```

If you are using a C shell, enter the following command instead:

```
setenv TERM motifterm
```

- Step 2: Place the AP in single user mode so that no other activities can use the AP during the full AP backup. A Y-NET reset occurs when you place the AP into single user mode. After ensuring that the AP is free for use, enter the following command:

```
init 1
```

It takes several minutes for the run level to change. You can use the **who -r** command to ensure that the run level has changed.

- Step 3: Start the **sysadm** menu operation:

```
sysadm
```

Refer to the *NCR UNIX SVR4 MP-RAS Administrator Guide: OA&M Menu Interface* for information about using the **sysadm** menus.

**NOTE:** Your AP window size must be 80x24 for the **sysadm** menus to display properly.

- Step 4: From the initial **sysadm** menu, choose the following selections:

```
Backup_Services
System
All
```

The system displays the following message. Note that although the message indicates a full system backup will not be performed, */etc/Backup* and */etc/Ignore* have been modified so that a full system backup will be performed.

**WARNING:** The system backup function you have selected does not backup a complete image of your system. To backup a complete image of your system refer to the SYSTEMS ADMINISTRATORS: ADVANCED TASKS manual.

Press ENTER to continue or DELETE to abort the backup



## Backing Up Information

- Step 5:** Press RETURN when you are ready. The following message appears:

Calculating approximate number of cartridge  
tape(s) required. Please wait.

After about 15 or 20 minutes, the system displays a message informing you how many tapes are required to back up the entire AP.

- Step 6:** Place the first tape in the tape device and press RETURN. The system begins the backup process and displays the following message, followed by a list of files being backed up:

Complete System backup in progress

Each file name appears on the screen as it is backed up to tape. When the backup completes, the following message is displayed:

The Complete Backup is now finished.

Do you want to verify the backup cartridge  
tape(s) (yes/no)?

- Step 7:** Enter yes to verify the backup. Verification takes about 15 to 20 minutes. When the verification completes, the following message appears:

System backup has completed successfully.  
Press enter to continue::

- Step 8:** Press RETURN. The **sysadm** menu appears.

**NOTE:** Remember to remove the backup tape, write protect it, and put it in a safe place. You will need this tape if you should have to restore your complete system as described in the “Restoring Information” chapter.

## Copying Files to NCR TOWERS

You can use the following procedure to copy files from an NCR System 3000 to one of the NCR TOWER machines.

---

### Procedure

Perform the following steps to copy files to NCR towers:

Step 1: Create the tape on the NCR System 3000 machine using the following commands:

```
find . -print > /tmp/backup list
cpio -ocvB -H odc < /tmp/backup list | \
dd of=/dev/rmt/device_name
```

See Appendix B, “Device Names and Numbers,” for an explanation of device names.

Step 2: Copy the files from the tape onto the NCR TOWER system using the following commands:

```
cd directory
dd if=/dev/rstp/0yy | cpio -icvB
```

## Retensioning a Cartridge Tape

You can use the following procedure to retension a cartridge tape. For more information, refer to **tapecntl(1M)** in the *Reference Manual*.

---

### Procedure

Perform the following command to retension or erase a cartridge tape:

To retension a cartridge tape, enter the following command:

```
tapecntl -t [-e] [-w] [-d device_name]
```

- The default device name is `/dev/rmt/c0t3d0s0`. You do not need to use the `-d` option unless the tape drive is not the default one.
- The `-e` option erases the data on the cartridge tape. When you have a tape you can read but not rewrite, erasing the data on the tape might solve the problem.
- The `-w` option rewinds the cartridge tape.

This chapter provides instructions for restoring information from backups.

|                                                  |      |
|--------------------------------------------------|------|
| Restoring File Systems .....                     | 7-2  |
| Restoring Individual Files and Directories ..... | 7-5  |
| Restoring Raw Devices .....                      | 7-8  |
| Restoring an Entire Disk .....                   | 7-10 |
| Restoring the Entire System .....                | 7-20 |
| Restoring a Corrupted Root File System .....     | 7-21 |
| Restoring an Entire NCR 3600 .....               | 7-27 |

## Restoring File Systems

Use this procedure to restore entire file systems or just selected files or directories from a file system.

---

### Before You Start

Any file system accessed during the restore must be mounted.

To restore a corrupted *root* , */stand* or */usr* file system refer to the “Restoring an Entire Disk” section in this chapter.

**NOTE:** Run level 1 does NOT prevent users from using the system. Be sure that users do not access the system while you are trying to restore files.

### Caution

If you do not want to restore all files unconditionally, do NOT specify the option **-u** for the **cpio** command!

---

### Procedure

Perform the following steps to restore a file system:

Step 1: Login as **root** and change to the root directory of the file system you are restoring:

```
cd directory
```

Step 2: Reset the user file size limit to *unlimited*. This will ensure that very large files which exceeds the size limit of a single file set by **ulimit** will nevertheless be restored.

```
ulimit unlimited
```

Step 3: Insert the first backup media (labeled **part 1**) in the appropriate drive and enter:

```
cpio -icvBdmul -I /dev/rmt/device_name \
["pathname"]
```

See Appendix B, “Device Names and Numbers,” for an explanation of device names.

The *pathname* field indicates the path name of the file to be restored and **MUST** be the same as the path name on the backup tape. For example, if you backup up */home* using the **find .** command, you must specify a relative pathname such as for example *./user1/myfile* to restore a single file. If you created a backup using the absolute pathnames, you must specify the */* at the beginning of the pathname to be restored.

- If the *pathname* field is not specified, all files on the tape are restored.
- To restore all files in a directory, use the asterisk. For example, to restore all files in the */sbin* directory, specify *sbin/\** for a relative pathname.

## Restoring Information

Step 4: If the file(s) specified in the *pathname* field are not on the backup media, the following messages are displayed.

```
End of medium on "input".
Change to part X and press RETURN key.
{q}
```

- To continue with the restore, insert the next backup media and press the ENTER key.
- To terminate the restore, enter the letter **q** (quit).

Step 5: Perform the **fsck** utility on the file system that contains the restored files. Remember to first unmount the file systems you need to check:

```
umount file_system
fsck -F FSType /dev/rdsk/device_name
```

The *device\_name* is the name of the slice where the file system to be checked physically resides.

Step 6: Reset the single file size limit to the original value by either logging out and log back in or by performing the following command:

```
ulimit num_of_orig_blocks
```

# Restoring Individual Files and Directories

Use this procedure to restore files or directories from a file system using the command line.

---

## Before You Start

Any file system accessed during the restore must be mounted.

To restore a corrupted root file system, refer to the root restore procedure later in this chapter.

**NOTE:** Run level 1 does NOT prevent users from using the system. Be sure that users do not access the system while you are trying to restore files.

---

## Procedure

Perform the following steps to restore individual files and directories:

Step 1: Login as **root** and change to the root directory of the file system you are restoring:

```
cd directory
```

Step 2: Reset the user file size limit to *unlimited*. This will ensure that very large files which exceeds the size limit of a single file set by **ulimit** will nevertheless be restored.

```
ulimit unlimited
```



## Restoring Information

Step 3: Insert the first backup media (labeled **part 1**) in the appropriate drive and enter:

```
cpio -icvBdmul -I /dev/rmt/device_name \
["pathname"]
```

See Appendix B, “Device Names and Numbers,” for an explanation of device names.

### Caution

The **cpio** option **-u** restores all files unconditionally.

The *pathname* field indicates the path name of the file to be restored and **MUST** be the same as the path name on the backup tape. For example, if you backup up */home* using the **find .** command, you must specify a relative pathname such as for example *./user1/myfile* to restore a single file. If you created a backup using the absolute pathnames, you must specify the **/** at the beginning of the pathname to be restored.

- If the *pathname* field is not specified, all files on the tape are restored.
- To restore all files in a directory, use the asterisk. For example, to restore all files in the */sbin* directory, specify *sbin/\** for a relative pathname.

Step 4: If the file(s) specified in the *pathname* field are not on the backup media, the following messages are displayed.

```
End of medium on "input".
Change to part X and press RETURN key. [q]
```

- To continue with the restore, insert the next backup media and press the ENTER key.
- To terminate the restore, enter the letter q (quit).

Step 5: Perform the **fsck** utility on the file system that contains the restored files. Remember to first unmount the file systems you need to check:

```
umount file_system
fsck -F fstype /dev/rdisk/device_name
```

The *device\_name* is the name of the slice where the file system to be checked physically resides.

Step 6: Reset the single file size limit to the original value by either logging out and log back in or by performing the following command:

```
ulimit num_of_orig_blocks
```

## Restoring Raw Devices

Use this procedure to restore raw devices.

---

### Before You Start

To prevent files from being modified during the restore, put the system in single user mode (run level s) or be sure no one is using the raw area you are restoring to.

For databases residing on raw slices, you can shutdown the database to make the data on the raw slices inaccessible to users.

Make sure no daemons are running which could write to the raw data slices. The following command displays the process IDs of the processes using the device:

```
fuser /dev/dsk/device_name
```

Backups of raw devices are done using the **dd** command. This means that the backup is a physical byte-by-byte backup of physical blocks and not a logical file-by-file backup as with for example the **cpio** command. Therefore, you cannot restore individual files or directories.

---

**Procedure**

For each raw slice, perform the following step:

Step 1: To restore a raw data slice, use the following command:

```
dd if=/dev/rmt/device_name bs=512k
of=/dev/rdisk/slice_name
```

For example, to restore the slice `/dev/rdsk/c0t0d0s4` from cartridge tape `/dev/rmt/c0t3d0s0`, type the following command:

```
dd if=/dev/rmt/c0t3d0s0 bs=512k
of=/dev/rdsk/c0t0d0s4
```

## Restoring an Entire Disk

---

### Why/When

If the information on one or more disks becomes corrupt or destroyed, you can recover the data provided you have the necessary backups of both the file system data AND the disk layout information. For each disk in your system, you must have a backup of the disk layout information on your maintenance diskette. If your maintenance diskette does not have enough room left, you can create a `s5` file system on another diskette, then copy the information to that diskette. When you need to retrieve the information, simply mount that additional flex diskette. For a mount-point, create a directory as for example `/mnt2`. It is a good idea to have a printed hard copy of the disk layout information.

---

### Before You Start

Before you begin the restore procedure, be sure of the following:

- You must know the file system type and mount point for each file system on the disk. Refer to the `vfstab` file you saved on the maintenance diskette for information about file system types and mount points. (This file should be in the `/<mount_point>/restore` directory.)
- Your maintenance diskette must also contain a backup copy of the `vtoc` (volume table of contents) structure that contains UNIX slicing information for the disk. Find the procedure for creating these backups in the section “Backing up File System and Disk Information” in this chapter.

- You must have a backup of each file system on the disk.
- You must have a backup of each raw data slice on the disk.

**NOTE:** Pathnames on the backup media must NOT begin with a slash (/). For example, use *usr/bin/filename* NOT */usr/bin/filename*. This type of backup is accomplished by changing to the *root* directory of the file system and performing the `find .` command. Refer to the section “Backing up File Systems” in Chapter 6.

## Overview of Restoring an Entire Disk

The tasks that will be described step by step to restore an entire disk are the following:

- Repartition and reslice the root disk in the maintenance file system.
- If the root disk is restored, the *root*, */usr*, */var*, and */stand* file systems are restored from the maintenance file system. All other file systems such as */home* or any user defined file systems are restored from single user mode.
- If a non-root disk is restored, you can restore all file systems (except */usr*) in multiuser mode as long as no users are accessing the area being restored.

**NOTE:** While the repartitioning and reslicing for the root disk must be performed from the maintenance file system, the repartitioning and reslicing of a non-root disk could also be performed from the single user or multiuser mode. But since you saved the slicing information for each disk on the maintenance diskette, it is easier to mount the flex and read the information when the system is running.

Also, restoring from maintenance file system is only done for the *root*, */usr* and */stand* file systems because those file systems are necessary to perform a reboot from hard disk. All other file systems are restored from the single user or multiuser mode instead of the maintenance mode simply for better performance.

---

## Reconfiguring a Disk

Perform “Access the Maintenance File System” only when you restore the root disk or when you restore the disk where */usr* resides.

### Access the Maintenance File System

Use the following procedure to access the maintenance file system:

- Step 1: Insert the first boot flex diskette into the flex disk drive.
- Step 2: Reboot the system from flex diskette.
- Step 3: When instructed, insert the second boot flex diskette into the flex disk drive and press the ENTER key.
- Step 4: When the following message appears, select maintenance by pressing 2:

Select one of the following:

- 1. Continue Installation
- 2. Perform System Maintenance
- 3. Perform System Restore
- 4. Perform Micro Channel Configuration

Type selection number, then press ENTER.>

**NOTE:** Menu selection 3 Perform System Restore only works if you restore from backup tapes created through the Open System Administrator menus or OSA commands. Refer to the *Open System Administrator Guide* for more information.

- Step 5: When the following message appears, select y to mount the maintenance floppy disk:

Would you like to mount the maintenance floppy disk (y)/n?

Step 6: When instructed, insert the maintenance flex diskette into the flex disk drive and press the ENTER key.

**NOTE:** Do NOT remove the maintenance flex diskette.

## Repartitioning and Reslicing the Disk

The following steps begin the restore procedure and must be completed whether you are restoring a root or non-root disk:

Step 1: Check to be sure the device node for the disk to be recovered exists:

```
ls -l /dev/rdisk/c0t5d0s0
```

Step 2: If the device node does not exist, find out the major and minor numbers and then create it:

```
nodes -d /dev/rdisk/c0t5d0s0
mknod /dev/rdisk/c0t5d0s0 35 80
```

Step 3: Partition the disk using the **fdisk** command. For example, enter:

```
fdisk /dev/rdisk/c0t5d0s0
```

If the disk is not 100% UNIX, manually enter the partitioning information that you saved to a file (such as *fdisk5*) in the maintenance file system. The file should be located in the */maint/restore* directory.

Even if the partitioning information displayed by **fdisk** is correct, you must enter the “Update disk configuration and exit” selection to partition the disk.

Step 4: Initialize the UNIX partition. For example, enter:

```
dklayout -pvd /dev/rdisk/c0t5d0s0
```



## Restoring Information

- Step 5: Update the volume table of contents (**vtoc**) with the slicing information for the UNIX partition. For example, enter the following command to recover information from the *c0t5d0s0* file:

```
edvtoc -f /maint/restore/c0t5d0s0
/dev/rdisk/c0t5d0s0
```

- Step 6: Check to be sure the device nodes for all the slices on the disk exist. If the device nodes do not exist, use the **nodes** command to find out the correct major and minor numbers and then use the **mknod** command to make the necessary nodes. You must create the block-special as well as the character-special device node for each slice. For example:

```
ls -l /dev/rdisk/c0t5d0s1
```

If the node for slice 1 does not exist:

```
nodes -d /dev/rdisk/c0t5d0s1
mknod /dev/dsk/c0t5d0s1 b 35 81
mknod /dev/rdisk/c0t5d0s1 c 35 81
```

---

## Restoring Data to the Root Disk

Perform the following steps to restore data to the root disk.

### Restoring the *root*, */stand*, */var*, and */usr* File Systems in Maintenance Mode

**NOTE:** You can NOT restore the */usr* file system in maintenance mode if it is a vxfs or Volume Manager controlled file system.

To bring your system up to single user mode, you need to restore at least the root file system, the */stand* file system and the */usr* file system. You must restore those file systems from the maintenance mode. It is recommended that you also restore the */var* file system from the maintenance mode to avoid any error messages at system startup time. Restore all other file systems remaining on the root disk from the single user mode.

Step 1: Set your PATH variable so that it includes the current directory:

```
PATH=. $PATH; export PATH
```

Step 2: Create the root file system using the information you saved on the maintenance diskette:

```
/etc/fs/ufs/mkfs [-o FSTypespecificoptions] \
/dev/dsk/c0t6d0s1 num_blocks
```

For example:

```
/etc/fs/ufs/mkfs nsect=54, ntrack=15, \
bsize=4096, fragsize=1024, cgsiz=16, \
free=10, rps=60, npi=2048, opt=t, apc=0, \
gap=0 /dev/dsk/c0t6d0s1 1125090
```

Step 3: Mount the root file system:

```
mount -F ufs /dev/dsk/c0t6d0s1 /mnt
```

Step 4: Use the **nodes** command to determine the major and minor numbers for the tape device node. For example:

```
/install/usr/sadm/sysadm/bin/nodes -d \
/dev/rmt/c0t3d0s0
```

Step 5: Create the tape device node, using the major and minor number returned by the **nodes** command. For example:

```
mknod /dev/rmt/c0t3d0s0 c 35 67
```

Step 6: Reset the user file size limit to *unlimited*.

```
ulimit unlimited
```

Step 7: Change to the mount point directory of the root file system to be restored on disk:

```
cd /mnt
```

Step 8: Restore the root file system on the disk by inserting the tape in the tape drive first and then entering the **cpio** command. For example:

```
cpio -lcvBdmu -I /dev/rmt/c0t3d0s0
```

## Restoring Information

Step 9: Change to the mount point directory of the stand file system to be restored on disk:

```
cd /mnt/stand
```

Step 10: Restore the stand file system on the disk by inserting the tape in the tape drive first and then entering the **cpio** command. For example:

```
cpio -icvBdmu -I /dev/rmt/c0t3d0s0
```

Be sure to use backups that do NOT begin path names with a slash (/). See the Note in the “Before You Start” section.

Step 11: Copy the boot program from your restored root file system to the *BOOT* slice 7 (blocks 1–28):

```
dklayout -b -f /mnt/etc/boot -d \
/dev/rdisk/c0t6d0s0
```

Perform Steps 12 through 16 if */usr* is a separate file system.

Step 12: Create *ich* */usr* file system using the information you saved on the maintenance diskette. For example:

```
/etc/fs/ufs/mkfs -o nsect=54, ntrack=15, \
bsize=4096, fragsize=1024, cgsiz=16, \
free=10, rps=60, npi=2048, opt=t, apc=0, \
gap=0 /dev/dsk/c0t6d0s3 1125090
```

Step 13: Make the root directory for the */usr* file system:

```
mkdir /mnt/usr
```

Step 14: Mount the */usr* file system:

```
mount -F ufs /dev/dsk/c0t6d0s3 /mnt/usr
```

Step 15: Change to the mount point directory of the */usr* file system to be restored on disk:

```
cd /mnt/usr
```

Step 16: Restore the */usr* file system on the disk by inserting the tape in the tape drive and then entering the **cpio** command. For example:

```
cpio icvBdmu -I /dev/rmt/c0t3d0s0
```

Perform Steps 17 through if */var* is a separate file system.

Step 17: Create the */var* file system using the information you saved on the maintenance diskette. For example:

```
/etc/fs/ufs/mkfs -o nsect=18, ntrack=9, \
bsize=4096, fragsize=1024, cgsiz=32, \
free=10, rps=60, npi=2048, opt=t, apc=0, \
gap=4 /dev/dsk/c0t6d0s4 399978
```

Step 18: Make the root directory for the */var* file system:

```
mkdir /mnt/var
```

Step 19: Mount the */var* file system:

```
mount -F ufs /dev/dks/c0t6d0s4 /mnt/var
```

Step 20: Change to the mount point directory of the */var* file system to be restored on disk:

```
cd /mnt/var
```

Step 21: Restore the */var* file system on the disk by inserting the tape in the tape drive and then entering the **cpio** command. For example:

```
cpio icvBdmu -I /dev/rmt/c0t3d0s0
```

Step 22: Unmount the file systems you restored and perform the **fsck** command. For example:

```
umount /mnt/usr
fsck -F ufs /dev/rdsk/c0t6d0s3

umount /mnt/var
fsck -F ufs /dev/rdsk/c0t6d0s4
```

```
umount /mnt/stand
fsck -F ufs /dev/rdsk/c0t6d0sa

umount /mnt
fsck -F ufs /dev/rdsk/c0t6d0s1
```

## Exit the Maintenance File System

Use the following procedure to exit the maintenance file system:

- Step 1: Exit the maintenance file system by entering CTRL-d. A message asking if you are finished with system maintenance is displayed. The message also says that all mounted file systems will be unmounted automatically.
- Step 2: Answer y. The system will then automatically start rebooting.
- Step 3: Remove the maintenance flex diskette from the drive.

**NOTE:** If your root file system contains the */usr* and */var* directory, the system should be able to come up into multiuser mode. However, if the */usr* and */var* file systems are separate file systems, your system will automatically only come up into single user mode after exiting the maintenance file system.

## Restoring the Remaining File Systems on the Root Disk from Single User Mode

Perform Step 1 only if your system is not already in single user mode:

- Step 1: Login as root and enter the single user mode:

```
init s
```

- Step 2: Mount the maintenance diskette to access the file system information:

```
mount -F s5 /dev/dsk/f0 /mnt
```

**NOTE:** Make sure the maintenance diskette is write-enabled. If the diskette is read-only protected, the **mount** command returns the error message “no such device.”

- Step 3: Re-create every file system residing on the root disk using the command that you saved on the maintenance diskette:

```
mkfs -F FSType [-o FSType specific options]
/dev/rdisk/device_name
```

- Step 4: Verify that all your file systems residing on the root disk and which are to be restored are mounted:

```
mount
```

If they are not mounted, make the root directory for the file system and mount each file system:

```
mkdir root-directory-of-filesystem
mount -F FSType /dev/dsk/device_name \
/mount_point
```

- Step 5: For each file system to be restored, change to the mount point directory of that file system:

```
cd /mount_point
```

- Step 6: For each file system on the disk, recover the data by inserting the tape in the tape drive and entering the following commands:

```
ulimit unlimited
cpio -icvBdmu -I /dev/rmt/device_name
```

- Step 7: Perform a system reboot so that all file systems will be checked for possible corruption.

```
cd /
shutdown -g0 -i6 -y
```

## Restoring the Entire System

You can use the following procedure to restore the entire system from the command line, if you backed it up using the procedure “Backing Up the Entire System”. If your system is an NCR 3600, follow the procedures in “Restoring an Entire NCR 3600.”

---

### Procedure

Perform the following steps to restore the entire system:

- Step 1: Follow the procedure “Restoring an Entire Disk”. Follow the steps given for restoring the *root* disk.
- Step 2: Repeat the procedure “Restoring an Entire Disk” for every non-root disk on the system. Follow the steps given for restoring a non-root disk.

## Restoring a Corrupted Root File System

If the *root* file system becomes corrupted, preventing the system from running, the following procedure may enable you to restore the file system.

**NOTE:** If you have OSA and a system backup made through OSA, you can restore your root file system using the OSA root restore flexes. Refer to the *Open Systems Administrator Guide* for the procedure.

---

### Before You Start

If the information on one or more disks becomes corrupt or destroyed, you can recover the data provided you have the necessary backups of both the file system data AND the disk layout information.



---

## Procedure

Perform the following steps to restore a corrupted root file system:

### Access the Maintenance File System

Use the following procedure to access the maintenance file system:

- Step 1: Insert the first boot flex diskette (volume 1) into the flex disk drive.
- Step 2: Reboot the system from flex diskette.
- Step 3: When instructed, insert the second boot flex diskette (volume 2) into the flex disk drive and press the ENTER key.
- Step 4: When the following message appears, select maintenance by pressing 2:

Select one of the following:

- 1. Continue Installation
- 2. Perform System Maintenance
- 3. Perform System Restore
- 4. Perform Micro Channel Configuration

Type selection number, then press ENTER.>

**NOTE:** Menu selection 3 Perform System Restore only works if you restore from backup tapes created through the “Open System Administrator” menus or OSA commands. Refer to the *Open System Administrator Guide* for more information.

- Step 5:** When the following message appears, select **y** to mount the maintenance floppy disk:

```
Would you like to mount the maintenance floppy
disk (y)/n?
```

- Step 6:** When instructed, insert the maintenance flex diskette (volume 3) into the flex disk drive and press the ENTER key.

**NOTE:** Do NOT remove the maintenance flex diskette.

## Restore the root File System

- Step 1:** When the shell prompt (#) appears, enter one of the following commands, depending on the type of file system for root.

```
/install/etc/fs/ufs/fsck -y
/dev/rdisk/device_name
```

– OR –

```
/install/etc/fs/s5/fsck -y
/dev/rdisk/device_name
```

See Appendix B, “Device Names and Numbers,” for an explanation of device names.

If you enter the wrong file system type, (for example, if *root* is a ufs file system and you enter s5), an error message appears and the file system is NOT checked. Try again, entering the other file system type.

## Restoring Information

**NOTE:** Some corrective actions by **fsck** result in some loss of data. You may be able to determine the amount and severity of data loss from the diagnostic output.

- If **fsck** is NOT successful and the file system can not be recovered, you must either reinstall and then restore the root file system from a backup OR you must recover the entire root disk. Refer to the book *Installing NCR UNIX SVR4* or to the “Maintenance and Recovery Techniques” chapter in this book for these procedures.
- If **fsck** is successful, continue with Step 8.

Step 2: Mount and restore the root file system. Use **ufs** or **s5** as appropriate for the mount command. For example:

```
cd /
mount -F ufs /dev/dsk/device_name /mnt
cd /mnt
```

See Appendix B, “Device Names and Numbers,” for an explanation of device names.

Step 3: Reset the user file size limit to *unlimited*. This will ensure that very large files which exceeds the size limit of a single file set by **ulimit** will nevertheless be restored.

```
ulimit unlimited
```

**Step 4:** Begin the restore by entering the following command, substituting the appropriate device name for your tape device:

```
/install/usr/bin/cpio -icvBdm -I \
/mnt/dev/rmt/device_name
```

See Appendix B, “Device Names and Numbers,” for an explanation of device names.

## Caution

If **fsck** indicated severe data loss and you want to restore all files from the root backup (even those with a modification date earlier than the existing files), include the letter **u** in the options for **cpio** (**-icvBdmu**).

When the end of the backup medium is reached, the following message appears:

```
End of medium on "input."
Change to part X and press RETURN key.[q]
```

- To continue with the restore, insert the next backup medium such as for example a cartridge tape and press the **ENTER** key.
- To terminate the restore, enter the letter **q** (quit).

**Step 5:** Perform the **fsck** utility on the file system that contains the restored files.

### **Exit the Maintenance File System**

- Step 1: Exit the maintenance file system by entering CTRL–d. This automatically unmounts all mounted file systems.
- Step 2: Answer y when asked whether you want to exit maintenance.
- Step 3: Remove the flex diskette from the drive. The system should reboot automatically.

You should now be able to use your hard disk to perform normal operations or to restore other file systems, directories, or files. If the system does not boot, you must either reinstall and then restore the *root* file system from a backup OR you must recover the entire *root* disk. Refer to the book *Installing NCR UNIX SVR4* or to the “Recovery Techniques” chapter in this book for these procedures.

## Restoring an Entire NCR 3600

Use this procedure to restore an entire NCR 3600 AP. You need the following to perform this procedure:

- A CRU disk (the same size as the AP base disk)
- An AP backup tape, created with the procedures in the “Backing Up Information” chapter
- AP boot disks/CD-ROM install
- A CD-ROM disk caddy

This procedure consists of the following tasks:

- Installing the AP CD-ROM
- Restoring the operating system

---

### Installing the CD-ROM

Refer to the *3600 Installation & Configuration Manual (Volume 1)* for the CD-ROM installation procedure. Before you can restore the backup tape to the disk, you need a bootable system; thus the need for the CD-ROM installation. You need only install a basic system, which does not need a specific configuration since the backup tape containing the configuration will be restored. Complete the following steps to install the CD-ROM:

Step 1: Remove all flex diskettes from the AP flex drive.

## Restoring Information

- Step 2:** At the AWS, select the appropriate AP icon (NOT the AP Cabinet, but the AP). From the Management menu, select the following menu options:

Subsystem Maintenance  
Sub System Reset  
Power Up Reset

Click on the APPLY and OK buttons. After the reset completes, select OK. This resets the AP and starts the boot process.

- Step 3:** After the boot process starts, open an AP window and monitor the boot process. In a few minutes, the system begins the hardware level 0 tests and then displays the following message:

To interrupt autoloading sequence, press the space bar within 5 seconds ...

- Step 4:** Press the space bar several times to access the Startup Subsystem Services main menu.
- Step 5:** Place the 3600 AP Boot Disk 1 of 2 into the flex drive.
- Step 6:** Select the following choices from the Startup Subsystem main menu:
7. System Boot Services  
4. System Processor Boot From Flex Disk
- Step 7:** When instructed, insert the flex labelled Boot Floppy 2 of 2 and press ENTER. The system asks whether you want to change the date.
- Step 8:** Enter **n** to continue installation.
- Step 9:** When the following message appears, enter **1** to continue installation:

```
Select one of the following:
 1. Continue Installation
 2. Perform System Maintenance
Type selection number, then press ENTER.>
```

**Step 10:** When the following message appears, enter **1** to continue installation:

```
WARNING: A new installation of the UNIX System
will destroy all files currently on the
system.
```

```
Select one of the following:
 1. Continue Installation
 2. Perform System Maintenance
Type selection number followed by ENTER.>
```

**Step 11:** When the system lists the default root disk, make sure it is **c0t6d0s0**. If this is not the default listed, change it when asked if you want to change the default root disk.

As the installation continues, the system asks whether you want to format the hard disk **c0t6d0s0**.

**Step 12:** Depending on the disk CRU that you used, you may or may not need to format the drive. Preferably, you do not since it can take quite some time. Typically, you should answer **n**.

The system then prompts you to insert the installation CD into the CD-ROM drive.

**Step 13:** Put the AP CD-ROM into the CD caddy, place the caddy into the CD-ROM unit, and press the RETURN key.

The operating system installation takes approximately 30 minutes. When the installation completes, the system prompts you to remove the diskette from the flex drive.



## Restoring Information

- Step 14: Remove the CD and any flex diskettes used during installation and press RETURN to reboot the system.
- Step 15: As soon as you press RETURN, close the AP window and wait approximately two minutes. Then open a window to monitor the reboot process.

---

### Restoring the Operating System

Perform the following steps to restore the system backup to disk.

Step 1: Login to the system as **root**. Use the password **ncr3600**.

Step 2: Change to the root directory:

```
cd
```

Step 3: Set **ULIMIT** to *unlimited* because of the large files that are to be restored:

```
ulimit unlimited
```

Step 4: Build the device node table:

```
#/usr/sadm/sysadm/bin/mktable>/dev/null 2>&1 &
```

**NOTE:** This command takes several minutes to complete. To verify that it has completed, enter the following command:

```
ps -ef | grep mktable
```

Step 5: Put the backup tape in the tape device and restore it as follows:

```
cpio -icvumld < /dev/rmt/device
```

Depending on the size of the system backup, the restore process could take several hours.

**Step 6:** Once the restore process is complete, reboot the AP:

```
init 6
```

Close the AP window as soon as you enter this command.

**Step 7:** After several minutes, open the AP window and monitor the boot process. When the AP is booted, log in and verify that the system is running properly.

# **Administering the UNIX SVR4 Mail Subsystem**

The purpose of this chapter is to help a system administrator take advantage of various options within the mail subsystem.

|                                                                            |      |
|----------------------------------------------------------------------------|------|
| Understanding SVR4 Mail .....                                              | 8-2  |
| Establishing a Smarter Host .....                                          | 8-5  |
| Establishing Domain Addresses .....                                        | 8-6  |
| Establishing a Mail Cluster or Gateway .....                               | 8-7  |
| Establishing Mail Service on a Networked File System<br>(RFS or NFS) ..... | 8-8  |
| Administering Alias Lists .....                                            | 8-11 |
| Other Surrogate File Routines .....                                        | 8-12 |
| Enabling SMTP .....                                                        | 8-14 |
| Stopping and Starting the SMTP Daemon .....                                | 8-17 |
| Disabling SVR4 Mail .....                                                  | 8-18 |
| Setting Up SMTP to Listen Over Multiple Networks ...                       | 8-20 |
| Reading the SMTP Log File .....                                            | 8-21 |

## Understanding SVR4 Mail

The SVR4 mail subsystem, new with System V Release 4, is the standard mail system of the System 3000. This implementation, sometimes called *mailsur*, replaces earlier UNIX mail systems based on the `sendmail(1M)` utility. (Though `sendmail(1M)` is still available under the BSD Compatibility package (*compat*), the SVR4 mail system is preferred because of its ease of configuration and overall robustness.)

Features of SVR4 mail subsystem include the following:

- Ease of configuration
- Support for UUCP as a mail transport
- Support for Simple Mail Transfer Protocol (SMTP) as a mail transport
- Extensibility to other mail transports
- Accessibility from a variety of mail user interfaces
- Support for Mail Exchangers (MX) under the Domain Name System (DNS)
- Alias files
- Mailing list directories
- Support for both UUCP-style and Internet-style addressing

By default, the mail subsystem provides electronic communications between users on the same machine, or between machines connected together on a UUCP network. The system administrator does not need to do anything for mail to work in the default manner.

---

## SVR4 Mail Components

The SVR4 mail programs are as follows:

- **mail(1), rmail(1)** — a simple user agent for sending or reading mail
- **mailx(1)** — a more advanced user agent
- **smtpqer(1M)** — called by **rmail(1)** to queue outbound mail for delivery via SMTP
- **smtpsched (1M)** — called by **smtpqer(1M)** and **cron(1M)** to schedule outbound mail for transmission. Called by **smtpd(1M)** during inbound mail delivery.
- **smtp(1M)** — called by **smtpsched(1M)** to transmit mail to a remote system via SMTP
- **smtpd(1M)** — the SMTP daemon listens on networks like TCP/IP and receives incoming mail messages via SMTP

For more information on each of the above utilities, see the corresponding manual page in the *NCR UNIX SVR4 MP-RAS Reference Manual*.

---

## Mail Administration Files

There are five files that are important to mail administration.

- **/etc/mail/maillsurr** — The mail surrogate file tells how to rewrite addresses and how to deliver messages through the networks. Described on the **maillsurr(4)** manual page.
- **/etc/mail/mailcnfg** — Permits various per-site options to be established. Described on the **mailcnfg(4)** manual page.
- **/etc/mail/namefiles** — This is the master alias path file that points to the alias file **/etc/mail/names**. Described on the **mailalias(1)** manual page.

- */etc/mail/names* — This file and */etc/mail/namefiles* (above) are used to define name mappings and address lists. Described on the **mailalias(1)** manual page.
- */usr/lib/mail/surrcmd/smtpcnfg* — When present, this file contains configuration information for the SMTP commands. Described on the **smtpcnfg(4)** manual page.

---

### Mail Addressing Styles

The default surrogate file contains entries to translate between domain style addresses and bang style addresses. Bang style addressing is characterized by exclamation points (also known as bangs) within the address. Examples are:

- *host!user*
- *host1!host2!user*

Domain style addressing is characterized by the use of the “at” ( @ ) sign. Examples are:

- *user@host.domain*
- *user@host*

## Establishing a Smarter Host

Although it is possible to maintain the data files for the UUCP network so that the system knows about hundreds or thousands of other systems that can be contacted, it is impractical to do so. It is often much easier to set up what is known as a “smarter host.” A “smarter host” is another UNIX system to which remote mail will be shipped; if the local machine doesn’t know about the system to which the mail is being sent.

For example, assume you need to send a mail message to *hosta!tony*, but your local machine does not know about *hosta*. The mail message can be automatically routed to the machine *worldly*, for example, which would have a more extensive list of UUCP connections.

Establishing a smarter host is done in two steps.

Step 1: Add a line to */etc/mail/mailcnfg* that says:

```
SMARTERHOST=smhost
```

where *smhost* is replaced with the name of the smarter system (*worldly*.)

Step 2: Remove the # (number) character from the beginning of the following line in */etc/mail/mailstarr*:

```
#'.+' '!(.+[!@%].+)' 'Translate R=!%X!\1'
```

The *SMARTERHOST* value is used while queuing outbound mail. If none of the mail transfer agents (for example, UUCP or SMTP) can immediately recognize the destination system, the mail is queued for the smarter host. The *smtpsched(1M)* utility optionally forwards mail to the smarter host on determining that it cannot contact the remote host directly.

## Establishing Domain Addresses

As distributed, mail knows about two forms of domain style addresses:

- *user@host*
- *user@host.UUCP*

It does not know about *user@host.domain*.

A domain name is an internationally recognized and registered name for a set of machines. Commercial entities may be registered under domain names similar to *.company-name.COM*. Educational entities may be registered under domain names similar to *.school.EDU*. (Note that *.UUCP* is not a true domain name. The high-level domain names of *.COM* and *.EDU* are assigned by a central authority.)

A system will generally know how to establish direct connections to other machines within the local domain, but will want to make use of a smarter host to take care of other domains.

To establish the local domain name, edit the file */etc/mail/mailcnfg* and add a line of the following form:

```
DOMAIN=domain
```

where *domain* is replaced with the domain name, such as *.company-name.COM* or whatever is appropriate. Any periods in the domain name will be converted to `\.` before being passed to the regular expressions in the surrogate file. The domain name is also used by the SMTP router when rewriting header files into RFC822 format.



## Establishing a Mail Cluster or Gateway

It is often desirable to assign a single name to a set (or a cluster) of machines by which all the machines in the cluster will be known to external machines, for purposes of mail. For example, a cluster of machines known internally under names such as *Xsysa*, *Xsysb* and *Xsysc*, could be assigned the cluster name *Xsys*. Mail sent from any of these machines would be shown as being from *Xsys*; that is, the internal names would not be known outside the cluster.

To establish a cluster name, add a line to */etc/mail/mailcnfg* that says:

```
CLUSTER=extname
```

where *extname* is the name by which the machine is known externally (*Xsys*.)

Defining the *CLUSTER* variable sets the default HELO name of the SMTP daemon *smtpd*(1M). It also sets the default ...remote from... information on the *From* header line of outgoing mail.

The cluster concept is usually employed on mail gateways to hide internal networks from the outside world. It is often useful to define aliases for internal users by creating mail alias databases on the gateway system. See the “Administering Alias Lists” section of this appendix and the *mailalias*(1) manual page for details.

## Establishing Mail Service on a Networked File System (RFS or NFS)

With the arrival of inexpensive Local Area Networking (LAN) and networked file systems such as RFS and NFS, clusters of machines that share many file systems can be set up. It is also possible to share */var/mail* across the machines. In this case, you can arrange to have all users' mailboxes created on only one machine, but accessible from all machines.

As an example, assume that you want the machines *Xsysa*, *Xsysb* and *Xsysc* to share the mail directory under *Xsysa*. In addition, the entire file system for each system is mounted under the names */Xsysa*, */Xsysb* and */Xsysc*. All users have home directories under file systems named */homea*, */homeb* and */homec*, which are mounted on the corresponding machines.

To establish a shared */var/mail* file system, complete the following steps:

- Step 1: Make certain that */var/mail* from *Xsysa* is advertised.
- Step 2: Remove the directory */var/mail/saved* from the systems that will not have a local */var/mail* (*Xsysb* and *Xsysc*.)

**Step 3:** Add a line to */etc/mail/mailcnfg* that says:

```
FAILSAFE=Xsysa
```

With this specified, mail will look for the presence of */var/mail/:saved*. If the directory is not there (indicating that the network connection to Xsysa has been lost,) mail will re-queue the file to be delivered Xsysa via other means (such as UUCP or SMTP).

**Step 4:** Move any mailboxes from */var/mail* on Xsysb and Xsysc to Xsysa (otherwise the files will be inaccessible.)

**Step 5:** Mount */var/mail* from Xsysa.

**Step 6:** To allow the *notify* program to identify where the user is logged in (so that it can notify the user when new mail arrives,) create a file on all machines named */etc/mail/notify.sys* with contents similar to the following:

```
Xsysa /Xsysa
Xsysb /Xsysb
Xsysc /Xsysc
```

The first column lists the name of the system and the second gives a pathname of the *root* file system for each machine.

## Administering the UNIX SVR4 Mail Subsystem

**Step 7:** To allow the *notify* program to handle a network failure, create a file on all machines named */etc/mail/notify.fsys* with contents similar to the following:

```
/homea Xsysa
/homeb Xsysb
/homec Xsysc
```

The first column lists a file system name and the second column contains the system (machine name) on which that file system is normally mounted. If *notify* cannot open the mail file for writing, it will look up the file system in this list and re-queue the file to be delivered to the corresponding system via other means, such as UUCP or SMTP.

## Administering Alias Lists

Before delivering a local mail message, mail will look up the user name to see if it has been aliased to another name or list of names. The master alias path file */etc/mail/namefiles* contains a list of files that mail will search for aliases. As distributed, this list contains only one file, */etc/mail/names*, to be searched for aliases. If the named alias is found at the beginning of a line within an alias file, the rest of that line will be used as the alias. This may contain a single name, or a list of names separated by white space. For example, if you want to set up a group mailing list, such as *andy.group*, that will be expanded, add a line similar to the following to the alias file:

```
andy.group tony paul john ned gary hailey mike
```

Recursive references are permitted, as in this reference to *andy.group* within another alias:

```
armida.dept andy.group danielle.group bob.group \
lee.group pier.group
```

Several alias files can be listed in *namefiles*, which may be kept anywhere on the machine. This permits different alias files to be owned by different administrators.

## Other Surrogate File Routines

---

### Logging Mail

Occasionally it may be necessary to keep a log of traffic going through the system. For example, if you were to write a program called `/usr/lib/mail/surrcmd/logmail` that takes three arguments (a log file name, the sender and the recipient,) it could log all external mail flowing through the system by using this surrogate entry:

```
' .+! .+' '.*' '> /usr/lib/mail/surrcmd/logmail \
/var/adm/mailtransport %R %n'
```

Another example would be to log traffic to or from a particular system (here to *xyz* and from *abc*):

```
'.*' 'xyz! .+' '> /usr/lib/mail/surrcmd/logmail \
/var/adm/mailto-xyz %R %n'
```

```
'abc! .+' '.*' '> /usr/lib/mail/surrcmd/logmail \
/var/adm/mailfrom-abc %R %n'
```

---

### Path Translation

Many systems have a path translation program available that will give the shortest route to a given system, based on various criteria or a database. An example of this is the public domain **smail** program. As an alternative to using a smarter host, the autorouter can be invoked as a final step in the *mailsurr* file:

```
' .+' '.*[!@].*' 'Translate R=|smail -A %n'
```

## Controlling Mail Resource Access

It is often necessary to control access to commercial Mail services. One method of doing this is to prevent any non-local users from sending mail to the commercial site using the **Accept** and **Deny** commands:

```
'[^!]+ 'attmail!.' 'Accept'
'.' 'attmail!.' 'Deny'
```

Another method is to use an external program to check the sender path to see if it is a valid user of the service. For example, this shell script returns 0 (zero) if the sender is a valid system and 1 (one) otherwise.

```
case "$1" in
 abc | def | ghi) exit 0 ;;
 *) echo "$1 is not permitted to send mail \
to external service"
 exit 1 ;;
```

If the script were installed as */usr/lib/mail/surrcmd/chksender*, it would be invoked as a delivery agent that will either continue or fail:

```
check senders more than one hop away
'.'!(.+)![^!]+ 'attmail!.' '< C=0;F=*; \
/usr/lib/mail/surrcmd/chksender\\1'
check senders one hop away
'(.+)![^!]+ 'attmail!.' '< C=0;F=*; \
/usr/lib/mail/surrcmd/chksender\\1'
```

## Enabling SMTP

The Simple Mail Transfer Protocol (SMTP) mail subsystem is delivered as a group of programs that allow UNIX system mail to send and receive mail using the SMTP protocol. This protocol is typically used over TCP/IP networks. However, as delivered, the SMTP processes can connect over any TLI-based, connection-oriented, transport that has been administered to have an SMTP service.

To establish SMTP service requires these steps:

- Step 1: By default, SMTP is installed in the *mailsur* (mail surrogate) file, but it is disabled. It can be enabled by removing the # (number sign) from the beginning of the following line in the *mailsur* file:

```
#'.+' '!(^[!])+!(.+) ' '< \
/usr/lib/mail/surrcmd/smtpqer -N %R \\1 \\2'
```

If your system has DNS installed, delete the `-N` option; if you wish to disable DNS queries, be sure to include the `-N` option.

Mail is addressed using the standard UNIX system mail formats of *host!user* or *user@host*. If *host* is known to support SMTP mail delivery, the mail will be queued for delivery using SMTP. If not, *smtpqer* will not accept the message, and delivery will be done by subsequent surrogates in the *mailsur* file.

All messages that are spooled for SMTP delivery are stored in the directory */var/spool/smtpq/host*, where *host* is the name of the machine to which mail is being sent.



**Step 2:** The list of machines that will accept SMTP mail is specified by the *netdird* service. Refer to the *netdird(1M)* manual page for information about adding services to this database. By default, the SMTP daemon *smtpd* will always start when your system is booted. If *smtpd* finds that there are no networks installed for which the SMTP service is defined, it will exit.

When the daemon *smtpd* receives a piece of mail, it does three things; 1. it inserts a valid UNIX system mail “From” header line; 2. it converts the recipient address to *host/user* form; and 3. it hands the message to *rmail* for delivery.

**Step 3:** The following entries must be added to the root *crontab* file. Refer to *crontab(1)* for an explanation of this file:

```
1,16,31,46 * * * * /usr/lib/mail/surrcmd/smtpsched
0 9 * * * /usr/lib/mail/surrcmd/smtpsched \
-c -w 1 -r 7
```

To do this, execute the following commands as root:

```
crontab -l > /tmp/cron.temp
ed /tmp/cron.temp
$
a
1,16,31,46 * * * * /usr/lib/mail/surrcmd/smtpsched
0 9 * * * /usr/lib/mail/surrcmd/smtpsched -c -w1 -r7
.
w
q
crontab /tmp/cron.temp
rm /tmp/cron.temp
#
```

The first new crontab entry specifies that mail that cannot be delivered immediately (as it is sent) is queued and re-tried at fifteen-minute intervals by *smtpsched*. You can change the interval by modifying the entry for *smtpsched* in root’s *cron* file.

**Smtpsched** also implements a back-off and retry algorithm which helps prevent failing mail destinations from interfering with others. See the **smtpsched(1M)** and **smtpcnfg(4)** manual pages for details.

SMTP logs all SMTP activity, including incoming mail messages, in the log file */var/spool/smtpq/LOG*. The second new crontab entry specifies that the SMTP log file is backed up once a day by *smtpsched*; log files from the previous day are located in */var/spool/smtpq/LOG.n*, where *n* is the day of the week (from 0 to 6.) The crontab entry specifies that the *smtpsched* program will also return undeliverable mail messages after seven days and warns about undelivered messages once a day. Refer to the **smtpsched(1M)** manual page.

## Stopping and Starting the SMTP Daemon

As delivered, the SMTP daemon starts automatically when the system enters a multiuser run level. The daemon is killed automatically when you shut down or go into single-user mode. You can also kill the daemon from the command line by entering the following command:

```
/etc/init.d/smtpd stop
```

If the daemon stops for any reason, your system will not be able to receive mail via SMTP. You can restart the daemon by entering the following command:

```
/etc/init.d/smtpd start
```

## Disabling SVR4 Mail

Though not recommended, it is possible to replace the SVR4 mail subsystem with other mail systems. Since mail systems are typically mutually exclusive, it would be necessary to disable the SVR4 mail subsystem first.

---

### Disabling the SVR4 SMTP Process

To disable SVR4 SMTP processes permanently, perform the following steps:

Step 1: Kill the SMTP daemon:

```
/etc/init.d/smtpd stop
```

Step 2: Remove the smtpd startup and kill scripts:

```
rm /etc/rc0.d/K74smtpd /etc/rc3.d/S88smtpd
rm /etc/rcp3.d/K35smtpd /etc/rc1.d/K35smtpd
rm /etc/rcp2.d/K35smtpd /etc/rc3.d/S88smtpd
rm /etc/rc2.d/S88.smtpd /etc/rcp2.d/S88smtpd
```

If necessary, you can restore these by creating links to */etc/init.d/smtpd*.

Step 3: Update the root's crontab file by commenting out any references to *smtpsched*:

```
crontab -l >/tmp/crontab.tmp
ed /tmp/crontab.tmp
1,$g/smtpsched/s/^/#/
w
q
crontab </tmp/crontab.tmp
rm /tmp/crontab.tmp
```

**Step 4:** Edit the file */etc/mail/maillsurr* and comment out any references to **smtpqer**:

```
ed /etc/mail/maillsurr
1,$g/smtpqer/s/^/#/
w
q
```

At this point, all SVR4 SMTP mail transfer processes are disabled.

---

### Disabling the User Agent

When you install an alternative mail system, you may avoid using the *maillsurr* translation scheme and mail transport selection by replacing the */usr/bin/mail* utility, which is linked to */usr/bin/rmail*. Doing so allows high-level interface programs such as **mailx** to continue in a uniform way, using the replacement **mail** and **rmail** programs.

## Setting Up SMTP to Listen Over Multiple Networks

*smtpd* will listen to any connection-oriented TLI network that provides the SMTP service. TLI networks are specified in */etc/netconfig*. For each network that is connection-oriented, *smtpd* will use *netdir\_getbyname(3)* to determine if the SMTP service exists for that network. If the service does exist, a port is opened at the address returned by this function. To make the listener listen to a new network, first administer the *netdir* databases, and then restart the listener.

## Reading the SMTP Log File

This section describes messages found in the current SMTP log file */var/spool/smtpq/LOG*, as well as in its daily backups with names of the form */var/spool/smtpq/LOG.n*, where *n* has a value from 0 through 6.

All log entries have the following general format:

*mm/dd hh:mm:ss utility message*

- *mm/dd* are the month and day the log entry was made.
- *hh:mm:ss* is the time the log entry was made (hours:minutes:seconds).
- *utility* is the name of one of the SMTP utilities (*smtp*, *smtpqer*, *smtpsched*, or *smtpd*).
- *message* is one of the messages listed later in this section.

On a busy system, attempting to follow the outcome of a single mail item is complicated by multiple simultaneous processes adding their own log entries.

---

### SMTPD Log Entries

The following messages may appear in the *message* field of the log entries:

#### **can't fork!!**

Indicates a system problem typically caused by having too many running processes or kernel tuning parameters for the number of processes configured too low.

**Connection refused:**

**smtpd** has refused an incoming connection on the basis of the configured **smtpd** load limiting and maximum number of **smtpd** process parameters. See the **smtpd(1M)** manual page to learn about these parameters.

**can't handle any more calls**

Indicates an **smtpd** bug; restart the **smtpd** program. If you are running NCR UNIX SVR4 MP-RAS Release 2.00.02, obtain the latest **PBASEUTIL** package to fix it.

**Maximum errors exceeded. Exiting.**

Indicates an **smtpd** bug; restart the **smtpd** program. If you are running NCR UNIX SVR4 MP-RAS Release 2.00.02, obtain the latest **PBASEUTIL** package to fix it.

**poll failed, errno = *errno***

Indicates network or other serious failure; restart the **smtpd** daemon.

**unexpected event *event\_code* on fd *file\_descriptor\_number***

Indicates network or other serious failure; restart the **smtpd** daemon.

**fdopen of socket for input**

File descriptor open failure.

**fdopen of socket for output**

File descriptor open failure.

**caller: DEBUG**

Logs an attempted **DEBUG** command in response to the **sendmail** virus.

**out of core!**

**smtpd** has failed to allocate memory and has exited.

**ERROR: can't find /etc/netconfig**

**smtpd** failed to open its transports while starting up and has exited.



**could not open any transports:**

**smtpd** failed to open its transports during start up and has exited.

**could not open transport: *TLI\_error\_string***

General TLI problems; *TLI\_error\_string* is an error string from the file */usr/include/sys/tiuser.h*. See **t\_error(3N)**.

**could not allocate t\_bind: *TLI\_error\_string***

General TLI problems; *TLI\_error\_string* is an error string from the file */usr/include/sys/tiuser.h*. See **t\_error(3N)**.

**could not bind address: *TLI\_error\_string***

General TLI problems; *TLI\_error\_string* is an error string from the file */usr/include/sys/tiuser.h*. See **t\_error(3N)**.

**transport *transport* opened on fd *fd\_nod* address *net\_address***

General TLI problems; see **t\_error(3N)**.

***transport*: finished connected *sss* seconds, *bbbb* bytes**

Indicates successful receipt of an e-letter into **smtpd**'s incoming mail queue. *transport* is typically **tcp**, *sss* is the number of seconds the transfer was active, and *bbbb* is the number of bytes received.

***transport*: bombed connected *sss* seconds, *bbbb* bytes**

Indicates unsuccessful receipt of an e-letter into **smtpd**'s incoming mail queue. *transport* is typically **tcp**, *sss* is the number of seconds the transfer was active, and *bbbb* is the number of bytes received.

**could not allocate t\_bind**

Indicates a TLI error; retried on successive incoming calls.

**push tirdwr failed**

Indicates a TLI error; retried on successive incoming calls.

**t\_listen failed**

Indicates a TLI error; retried on successive incoming calls.

*status count=running/maxrun, load=load/loadlim idled*

General **smtpd** status entry; does not indicate a problem. *maxrun* is the **smtpd -l** value, *running* is the number of **smtpd** child processes running, *loadlim* is the **smtpd -L** value, and *load* is **smtpd**'s impression of the system load to compare to *loadlim*. *status* may be any of the values in Figure 8-1.

| Status   | Description                                                                                                                    |
|----------|--------------------------------------------------------------------------------------------------------------------------------|
| accepted | The <b>smtpd</b> parent has accepted an incoming connection and is about to spawn a child process to accept the incoming mail. |
| refused  | Generally accompanies the "Connection refused:" message.                                                                       |
| startup  | <b>smtpd</b> has started but has not yet opened transports.                                                                    |
| exiting  | <b>smtpd</b> is exiting.                                                                                                       |
| incr     | The administrator has sent <b>smtpd</b> a SIGUSR2 signal, incrementing <i>maxrun</i> .                                         |
| decr     | The administrator has sent <b>smtpd</b> a SIGUSR1 signal, decrementing <i>maxrun</i> .                                         |
| idled    | The administrator has sent <b>smtpd</b> a SIGHUP signal, requesting it to exit gracefully after the last child process exits.  |
| unidled  | The administrator has sent <b>smtpd</b> a successive SIGHUP signal, cancelling a previous request to go idle.                  |

Figure 8-1: Status Values

***transport: status count=running/maxrun, load=load/loadlim idled***

General **smtpd** status entry; does not indicate a problem. *maxrun* is the **smtpd -l** value, *running* is the number of **smtpd** child processes running, *loadlim* is the **smtpd -L** value, *load* is **smtpd**'s impression of the system load to compare to *loadlim*, and *transport* is the name of the transport type (typically **tcp**). *status* may be any of the values in Figure 8-1.

## **SMTPSCHED Log Entries**

**Smtpsched** entries typically show opportunities to transfer mail to another system.

### **forwarding *control\_file* to SMARTERHOST**

**smtpsched** has determined that further attempts to send the mail from the local system are futile. The mail will be forwarded to the smarter host defined in */etc/mail/mailcnfg*. See **smtpcnfg(4)** for conditions which govern this file.

### **can't get memory for internal table**

Indicates an internal problem; this instance of **smtpsched** exits.

### **can't chdir to */var/spool/smtpq***

Be sure that the SMTP spool directory exists and is owned by **uucp**.

### **couldn't read */var/spool/smtpq***

Be sure that the SMTP spool directory exists and is owned by **uucp**.

### **couldn't lock *mailqueue***

A normal situation indicating that another instance of **smtpsched** is already servicing the mail queue *mailqueue*. A mail queue is a subdirectory under */var/spool/smtpq*. This message normally occurs if you send multiple messages to the same destination.

**couldn't read directory *mailqueue***

Be sure that the *mailqueue* subdirectory under */var/spool/smtpq* is owned by *uucp*.

***mailqueue* empty**

The nightly **smtpsched -c** has been called from crontab. The empty *mailqueue* is removed.

***mailqueue/workfile* inconsistent**

The nightly **smtpsched -c** has been called from crontab. A workfile without a corresponding data file (or vice-versa) will be removed.

***mailqueue/workfile* too old**

The nightly **smtpsched -r** has been called from crontab. The sender will be warned by mail.

**ignore *mailqueue/workfile*: not time yet**

**smtpsched -v** has been called from crontab. The given message is temporarily skipped due to **smtpsched**'s backoff and retry algorithm.

**returnmail *workfile* to *reply\_address* about *destination***

Mail is being returned to the local user due to some failure in delivery.

**dormail *mailqueue/workfile***

Routine log entry when **smtpsched** attempts delivery of a message from an incoming **smtpd** queue to a local user.

**success**

Routine log entry when **smtpsched** has delivered an incoming message to a local user via **rmail**. May also follow a successful **dosmtp** message, indicating a successful delivery of an outgoing message to a remote system.

**failure**

Routing log entry when **smtpsched** has failed to deliver a message to a local user via **rmail**. May also follow a failing **dosmtp** message, indicating a failing delivery of an outgoing message to a remote system.

**dosmtp *mailqueue/workfile***

**smtpsched** is calling **smtpd** to try to send a message to a remote system. A "success" or "failure" message may follow.

**passed *destination (ss sec)***

**smtpsched** is skipping the destination since it has consumed more than the configured **MXTPERD** time. See **smtpcnfg(4)**.

**fail *nn***

An **smtp** attempt has failed. The error codes *nn* have the meanings in Figure 8-2.

| Code | Description                                                                                                                                        |
|------|----------------------------------------------------------------------------------------------------------------------------------------------------|
| 64   | Command line usage error on local <b>smtp(1M)</b> .                                                                                                |
| 67   | Addressee unknown, syntax error in address, or data format error. Rejected by the other system's <b>smtp(1M)</b> .                                 |
| 68   | Host name unknown. Rejected by the other system's <b>smtpd</b> .                                                                                   |
| 69   | Service unavailable. Host not responding, or no route to host. This condition can trigger forwarding to the smarter host. See <b>smtpcnfg(4)</b> . |
| 70   | Internal software error.                                                                                                                           |
| 72   | Critical operating system file missing (for example, <i>/etc/services</i> ).                                                                       |
| 74   | IO error, but possible network gateway problem. This condition can trigger forwarding to the smarter host. See <b>smtpcnfg(4)</b> .                |
| 75   | Temporary failure; <b>smtpsched(1M)</b> retries.                                                                                                   |

Figure 8-2: Error Codes

---

## SMTPQER Messages

The following message is generated by **smtpqer**.

**queued message for host *host* user *user***

**smtpqer** has accepted the message onto its internal queues for outbound delivery to *host* and *user*.

---

## SMTP Messages

The following messages are generated by **smtp**.

**can't t\_open transport**

Typically retried.

**can't bind address for client**

Typically retried.

**can't alloc for client**

Typically retried.

**push tirdwr failed**

Typically retried.

**t\_connect failed**

This commonly-occurring error indicates that the destination is off-line. It is retried when cron calls **smtpsched**.

This appendix contains the following information:

|                                 |      |
|---------------------------------|------|
| The File System Hierarchy ..... | A-2  |
| Symbolic Links .....            | A-5  |
| File System Organization .....  | A-6  |
| File System Types .....         | A-8  |
| The s5 File System Type .....   | A-10 |
| The ufs File System Type .....  | A-17 |
| The bfs File System Type .....  | A-23 |
| The cdfs File System Type ..... | A-27 |

# The File System Hierarchy

The UNIX operating system was originally created for use on stand-alone computers. In today's networking environment, files need to be shared by machines with similar and dissimilar CPU architecture. The physical location of files and directories has been changed in UNIX System V Release 4 to define a standard organization for all UNIX machines and to facilitate sharing files in a network environment.

The degree to which files can be shared across a network depends on the type of information in the files (characteristics) and the CPU architecture of the machines.

The files (and directories) can be categorized in three ways:

## **Machine private files**

These are files that can not or should not be shared with other machines (for example, accounting logs and rc scripts). The *root* file system (*/*) contains machine private files.

## **Architecture dependent files**

These are files that are shareable among machines of the same CPU type (for example, binary executables). The */usr* file system contains architecture dependent files.

## **Architecture independent files**

These are files that are shareable among machines regardless of CPU type (for example, ASCII databases and on-line manual pages). The */usr/share* directory contains architecture independent files.

Files with different characteristics have been placed in different subtrees within the file system hierarchy.



## The Root File System

The root file system in UNIX System V Release 4 contains several new directories. The boxes in Figure A-1 indicate directories that are frequently made into separate file systems. Although */proc* is a separate file system, it resides in memory and does not require a separate disk slice. On the other hand, */stand* is automatically made into a separate file system and placed in slice "a" during install.

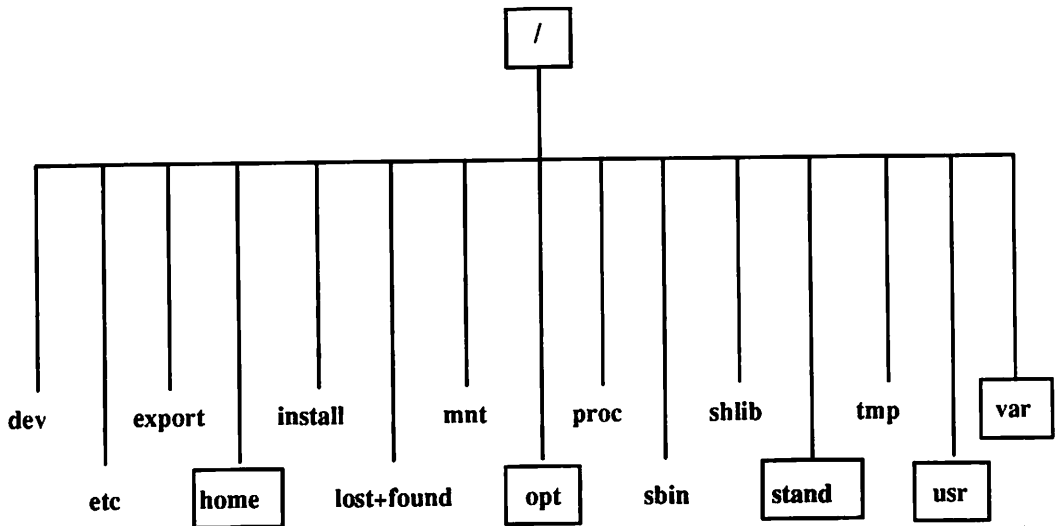


Figure A-1: The Root File System

**NOTE:** Entries such as */bin*, */lib* and */unix* appear if you perform an *ls* on the root directory; however, these entries are only symbolic links to the new locations (*/usr/bin*, */usr/lib*, and */stand/unix*).

Figure A–2 describes the contents of the directories in the root file system.

| The root (/) File System |                                                                                                                                                                    |
|--------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Directory                | Contents                                                                                                                                                           |
| <i>/dev</i>              | Character and block special files.                                                                                                                                 |
| <i>/etc</i>              | Machine–specific administration configuration files and system administration databases but no executables.                                                        |
| <i>/export</i>           | Default root of the exported file system.                                                                                                                          |
| <i>/home</i>             | Default root of subtree for user directories, formerly <i>/usr/acct</i> .                                                                                          |
| <i>/install</i>          | Directory for install scripts used during system installation.                                                                                                     |
| <i>/lost+found</i>       | Directory used by <b>fsck</b> for unreferenced files and directories.                                                                                              |
| <i>/mnt</i>              | Default temporary mount point for file systems.                                                                                                                    |
| <i>/opt</i>              | Default root of a subtree for add–on application packages. This directory was formerly <i>/appl</i> .                                                              |
| <i>/proc</i>             | Root of subtree for a special type of file system, the process file system, which is a mechanism for accessing the address space of a running process.             |
| <i>/sbin</i>             | Executables that are essential for booting and manual recovery. Some of these administrative executables are also in <i>/usr/bin</i> or <i>/usr/sbin</i> .         |
| <i>/shlib</i>            | Used for shared libraries.                                                                                                                                         |
| <i>/stand</i>            | Default root of the new boot file system (type bfs). This must be in slice a (slice is equivalent to partition in previous releases of the operating system).      |
| <i>/tmp</i>              | Small, system–generated, temporary files. This directory is cleared when the system enters multi–user mode.                                                        |
| <i>/usr</i>              | Root of a standard file system for static, shareable files. Only <i>/usr/share</i> contains architecture independent files; the others are architecture dependent. |
| <i>/var</i>              | Root of a subtree for varying files such as log files.                                                                                                             |

Figure A–2: Root File System Contents

## Symbolic Links

In addition to the traditional hard links that can link two or more files within a single file system, UNIX System V Release 4 provides symbolic links. A symbolic link is a file that contains the path name of another file or directory and acts as a pointer to that location. The kernel converts references to the symbolic link into references to the target file or directory.

Since you can symbolically link directories as well as files, you can rearrange the logical structure of a system's files without changing their physical location. Symbolic links also operate across file systems, even those residing on different machines.

### Displaying Symbolic Links

You can display symbolical links using the `ls` command with the `-l` option. The following examples show the listing for directories and files.

**Example 1: a symbolically linked directory**

```
ls -l /bin
lrwxrwxrwx 1 bin bin 8 Jul 13 15:31 /bin -> /usr/bin
```

**Example 2: a symbolically linked file**

```
ls -l /unix
lrwxrwxrwx 1 root root 11 Jul 23 14:45 /unix -> \
/stand/unix
```

**Example 3: a directory containing a symbolically linked file**

```
ls -l /
lrwxrwxrwx 1 bin bin 8 Jul 13 15:31 /bin -> /usr/bin
drwxrwxr-x 15 root sys 3584 Jul 25 08:21 dev
drwxrwxr-x 29 root sys 3072 Jul 25 05:54 etc
.
.
.
```

## File System Organization

A primary function of the UNIX operating system is to support file systems. In the UNIX system a file is a string of bytes with no other structure implied. Files are attached to a hierarchy of directories. A directory is merely another type of file that the user is permitted to use, but not to write; only the operating system can write directories. The combination of directories and files make up a file system. Figure A-3 shows the relationship between directories and files in a UNIX file system. The circles represent directories.

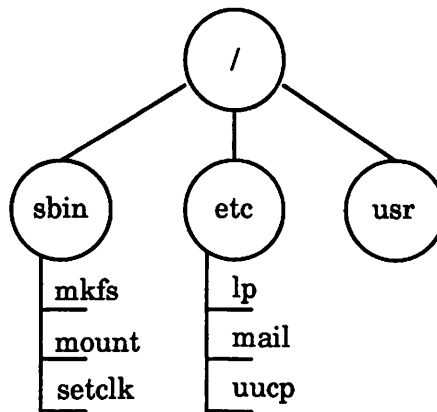


Figure A-3: A UNIX File System

The starting point of any UNIX file system is a directory that serves as the root of that file system. Somewhat confusingly, in the UNIX operating system there is always one file system that has the name root. Traditionally, the root directory of the root file system is represented by a single slash (/). The file system diagrammed in Figure A-3 is a root file system. If we mount another file system onto the root file system at a directory called */usr*, the result can be illustrated by the diagram in Figure A-4.

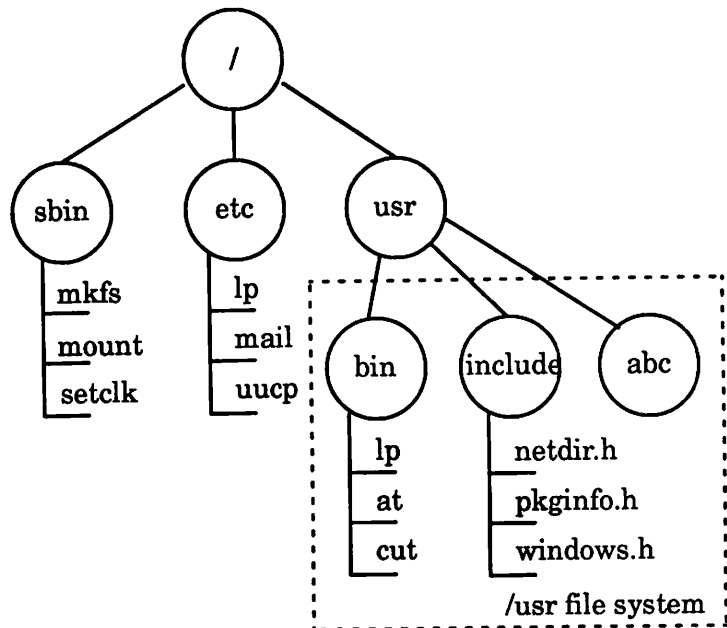


Figure A-4: Adding the /usr File System

A directory such as */usr* that is used to form the connection between the root file system and another mountable file system is sometimes called a “leaf” or “mount point.” Regardless of the term used, such a directory is the root of the file system that descends from it. The name of that file system is the name of the directory. In our example the name of the file system is */usr*.

Figure A-3 and Figure A-4 may be convenient representations of the file and directory structure of file systems, but they are not a particularly accurate or helpful way of illustrating how the UNIX operating system views a file system. The sections that follow describe file system types (FSTypes) as they appear to the operating system.

## File System Types

The following file system types are defined for use. In addition, you can define file system types of your own.

### **s5**

The traditional UNIX System V file system; supports data storage in logical blocks of 512, 1K, and 2K bytes.

### **ufs**

An implementation of the BSD “fast” file system; stores data in blocks of 4K or 8K and supports all System V file operations.

### **rfs**

Remote File Sharing (RFS) is re-implemented under VFS as a file system type. It supports file sharing among UNIX systems (distributed but not supported).

### **nfs**

Network File Sharing (NFS) was originally developed by Sun Microsystems. It supports file sharing among systems with different architectures and operating systems.

### **/proc**

A special file system that is used to access the address space of a running process. It is useful for debugging processes.

### **fifofs**

A file system type that is used for pipe files. It is not mountable.

### **specfs**

A file system type that is used for special files or devices. It is not mountable.

**bfs**

A file system type that is used for file system independent booting. A bfs file system contains all files necessary for booting and manual recovery.

**cdfs**

A read-only file system that is used specifically for CD-ROM (Compact Disc – Read Only Memory) devices.

**vxfs**

Also known as the Journaling File System, this file system type maintains a log of all transactions, reducing the time required to check and repair file systems. See the *Journaling File System Administrator Guide* for more information.

## The s5 File System Type

The operating system views an s5 file system as an arrangement of addressable blocks of disk space that belong to one of four categories:

- Block 0 (the boot block)
- Block 1 (the super-block)
- A variable number of blocks comprising the i-list
- A variable number of storage blocks, most containing data, some containing the free list and indirect addresses

This layout is illustrated in Figure A-5.



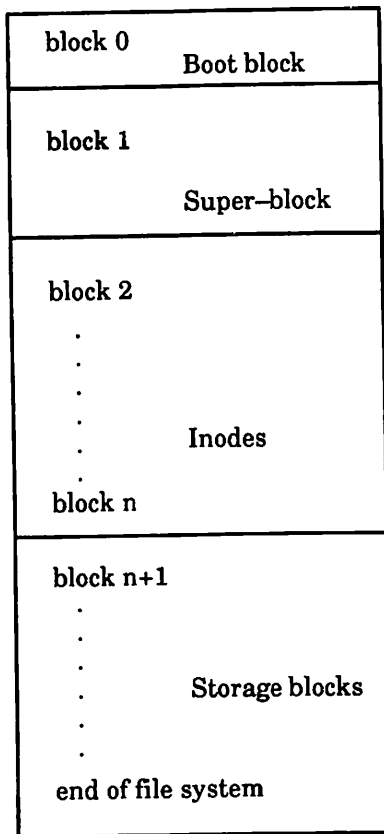


Figure A-5: The UNIX View of an s5 File System

---

### The s5 Boot Block

Although considered to be part of the file system, the boot block is not actually used by it. It is reserved for storing procedures used in booting the system. However, not all file systems are involved in booting. When a file system is not to be used for booting, the boot block is left unused.

---

### The s5 Super-Block

Much of the information about the file system is stored in the super-block, including such things as:

#### File system size and status

- Label (file system name)
- Size in logical blocks
- Read-only flag
- Super-block modified flag
- Date and time of last update

#### Inodes

- Total number of inodes allocated
- Number of free inodes
- Array of 100 free inode numbers
- An index into the array of free inode numbers

### **Storage blocks**

- Total number of free blocks
- Array of 50 free block numbers
- An index into the array of free block numbers

Note that the super-block does not maintain complete lists of free inodes and free blocks, but only enough to meet current demands as the file system is used. At almost any time, unless the file system is close to running out of inodes and storage blocks, there are sure to be more free inodes and free blocks than are listed in the super-block.

---

### s5 Inodes

The term “inode” (or i-node) stands for information node. A list of inodes is called an i-list and the position of an inode in an i-list is called an i-number.

An inode contains all the information about a file except its name, which is kept in a directory. Inodes are 64 bytes long, so there are 8 of them in a physical block. The length of an i-list is not fixed; it depends on the number of inodes specified when the file system is created. Specifically, an s5 inode contains:

- The type and mode of the file —the type can be regular, directory, block, character, symbolic link, or FIFO, also known as named pipe; the mode is the set of read–write–execute permissions
- The number of hard links to the file
- The user–id of the owner of the file
- The group–id to which the file belongs
- The number of bytes in the file
- An array of 13 disk block addresses
- The date and time the file was last accessed
- The date and time the file was last modified
- The date and time the file was created

The array of 13 disk block addresses is the heart of the inode. The first 10 are direct addresses; that is, they point directly to the first 10 logical storage blocks of the contents of the file. If the file is larger than 10 logical blocks, the 11th address points to an indirect block, which contains direct addresses instead of file contents; the 12th address points to a double indirect block, which contains addresses of indirect blocks. Finally, the 13th address in the array is the address of a triple indirect block, which contains addresses of double indirect blocks. Figure A-6 illustrates this chaining of address blocks stemming from the inode.

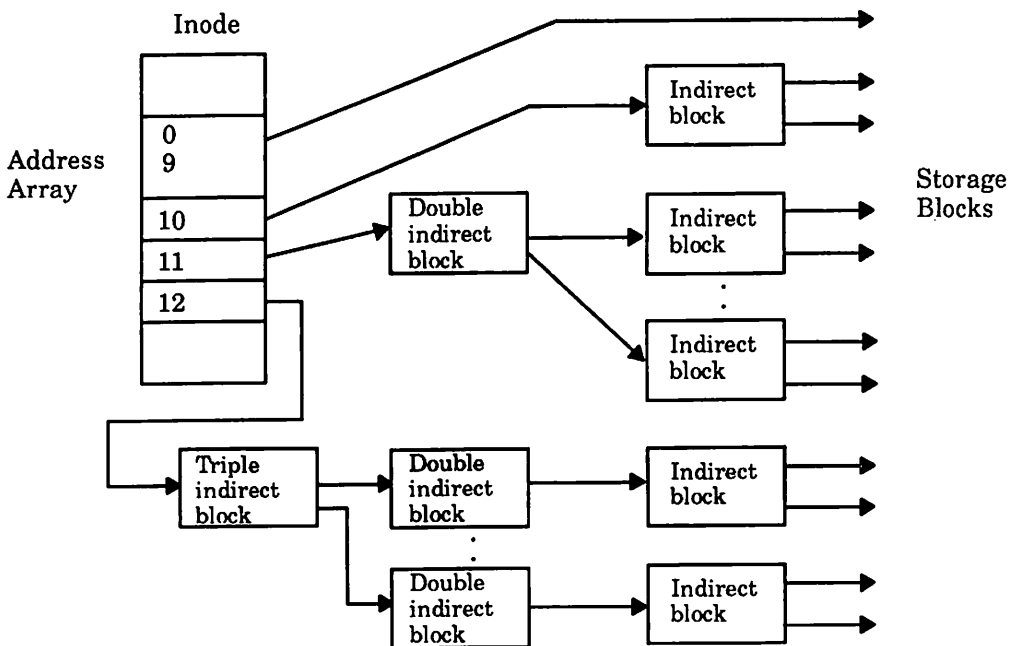


Figure A-6: The File System Address Chain for s5

---

## **s5 Storage Blocks**

The rest of the space allocated to the file system is occupied by storage blocks, also called data blocks. For a regular file, the storage blocks contain the contents of the file. For a directory, the storage blocks contain 16-byte entries. Each entry represents a file or subdirectory that is a member of the directory. An entry consists of 2 bytes for the i-number and 14 bytes for the filename of the member file or subdirectory.

---

## **s5 Free Blocks**

Blocks not currently being used as inodes, as indirect address blocks, or as storage blocks are chained together in a linked list of free blocks. Each block in the list carries the address of the next block in the chain.

## The ufs File System Type

The ufs FSType is considerably more complex in its design than the s5 FSType. In addition to the four categories of addressable blocks found in s5, there are several additional information management disk areas. There is also a radically different method of allocating and managing these blocks. Of primary interest is the fact that multiple super-blocks are made during the mkfs procedure. One of the replicas is stored in each cylinder group, offset by a certain amount. For multiple platter disk drives, the offsets are calculated so that a super-block appears on each platter of the drive. So if the first platter is lost, an alternate super-block can be retrieved. For platters other than the top one in a pack, the leading blocks created by the offsets are reclaimed for data storage.

Kept with the super-block is a summary information block. This block is not replicated, but is grouped together with the first super-block, normally in cylinder group 0. This summary block is used to record changes that take place as the file system is used, and lists the number of inodes, directories, fragments, and blocks within the file system.

Another feature found in ufs is the cylinder group map. This is a block of data found in each cylinder group that records the block usage within the cylinder. This information is kept directly following the super-block copy for that cylinder group.

To give an idea of the appearance of a typical ufs file system, Figure A–7 shows a series of cylinder groups in a generic ufs file system:

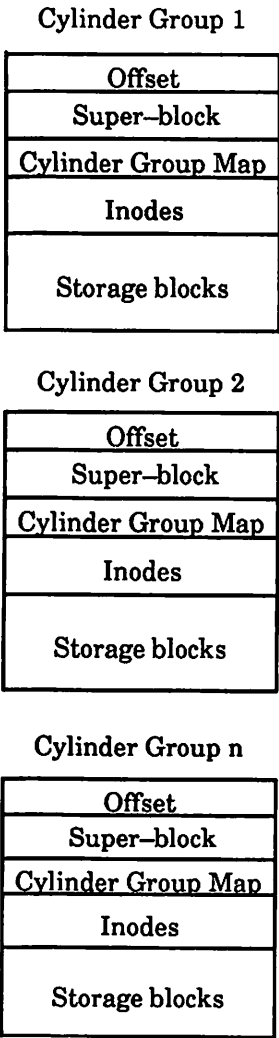


Figure A–7: The UNIX View of a ufs File System



---

## The ufs Boot Block

The boot block appears only in the first cylinder group (cylinder group 0) and is the first 8K in a partition. It is reserved for storing the procedures used in booting the system. If a file system is not to be used for booting, the boot block is left blank.

---

## The ufs Super-Block

Much of the information about the file system is stored in the super-block. A few of the more important things it contains are:

- The size and status of the file system
- The label (file system name)
- The size of the file system in logical blocks
- The date and time of the last update
- The cylinder group size
- The number of data blocks in a cylinder group
- The summary data block

---

### ufs Inodes

The inode information is kept in the cylinder information block. An inode contains all the information about a file except its name, which is kept in a directory. An inode is 128 bytes long. One inode is created for every 2K of storage available in the file system. This parameter can be changed when **mkfs** is used to create the file system, but it is fixed thereafter. A ufs inode contains:

- The type and mode of the file—the type can be regular, directory, block, character, symbolic link, or FIFO, also known as named pipe; the mode is the set of read–write–execute permissions
- The number of hard links to the file
- The user–id of the owner of the file
- The group–id to which the file belongs
- The number of bytes in the file
- An array of 15 disk block addresses
- The date and time the file was last accessed
- The date and time the file was last modified
- The date and time the file was created

The array of 15 disk addresses is the heart of the inode. The first 12 are direct addresses; that is, they point directly to the first 12 logical storage blocks of the contents of the file. If the file is larger than 12 logical blocks, the 13th address points to an indirect block, which contains direct addresses instead of file contents; the 14th address points to a double indirect block, which contains addresses of indirect blocks. The 15th address is unused.

Figure A–8 illustrates this chaining of address blocks stemming from the inode.

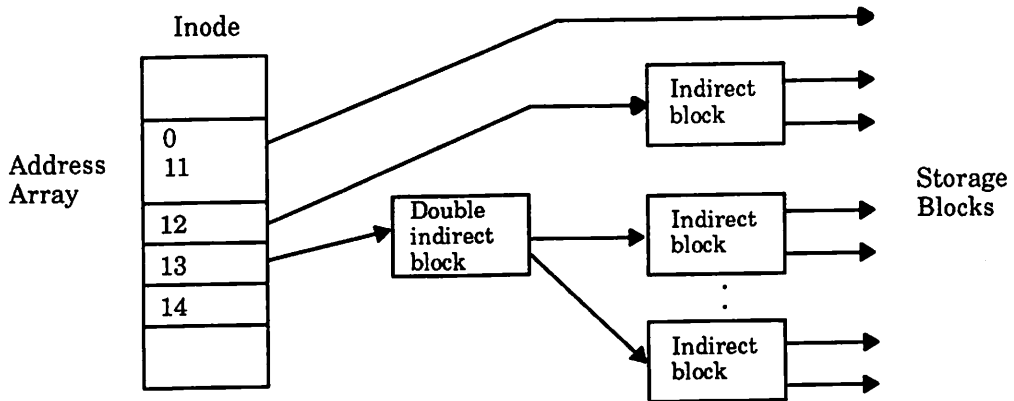


Figure A-8: The File System Address Chain in a ufs File System

## ufs Storage Blocks

The rest of the space allocated to the file system is occupied by storage blocks, also called data blocks. The size of these storage blocks is determined at the time a file system is created, and can be either 4096 or 8192 bytes. Because of these large block sizes, and the potential for waste with small files, ufs also has a subdivision of a block called a fragment. When a file system is created the fragment size may be set to 512, 1024, 2048, or 4096 bytes. Fragments of 1024 bytes are the most commonly employed. For a regular file, the storage blocks contain the contents of the file. For a directory, the storage blocks contain entries that give the inode number and the filename. ufs file names may be up to 255 bytes long. Each entry represents a file or subdirectory that is a member of the directory.

---

## **ufs Free Blocks**

Blocks not currently being used as inodes, as indirect address blocks, or as storage blocks are marked as free in the block map kept in the cylinder group summary information block. This block also keeps track of fragments in order to prevent fragmentation from degrading disk performance excessively.

## The bfs File System Type

The bfs FSType is a special-purpose file system. It contains all the stand-alone programs (for example, `unix`) and all the text files necessary for the boot procedures.

The object of the bfs FSType is to allow quick and simple booting. It is for this reason that bfs was designed as a contiguous flat file system. The only directory bfs supports is the *root* directory. Users may only create regular files; no directories or special files can be created in the bfs file system.

A bfs file system consists of three parts: the disk super-block, the inodes, and the storage or data blocks. The layout is illustrated in Figure A-9:

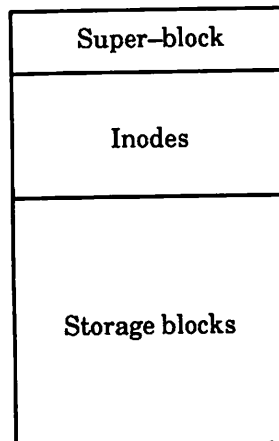


Figure A-9: The UNIX View of a bfs File System

---

## The bfs Super-Block

The following information is stored in the super-block:

- The magic number
- The size of the file system
  - The offset to the start of file system data (in bytes)
  - The offset to the end of file system data (in bytes)
- The sanity words.
  - There are four words used to promote sanity during compaction. They are used by the **fsck** command to recover if there has been a system crash at any time during the process of compaction.
  - See “Compaction” in this chapter for more information.

---

## bfs Inodes

The inode contains all the information about a file except its name. File names are kept in the root directory, the only directory in the bfs file system. An inode is 64 bytes long. The number of inodes is defined when **mkfs** is used to create the file system. An inode contains:

- The inode number. By convention this field is set to zero to indicate that the inode is available.
- The first data block
- The last data block
- The disk offset to the end-of-file (in bytes)
- The file attributes
  - The type and mode of the file
  - The user-id of the owner of the file
  - The group-id to which the file belongs
  - The number of hard links to the file
  - The date and time the file was last accessed
  - The date and time the file was last modified
  - The date and time the file was created

---

### bfs Storage Blocks

The remainder of the space allocated to the file system is taken up by storage blocks, also called data blocks. The size of the storage blocks is 512 bytes. The storage blocks are used to store the root directory and the regular files. For a regular file, the storage blocks contains the contents of the file. For the root directory, the storage blocks contain 16-byte entries. Each entry represents a file and consists of 2 bytes for the i-number and 14 bytes for the file name.

---

### Managing Data Blocks

The data or storage blocks for a file are allocated contiguously. The data block after the last data block used in the file system is considered the next data block available to store a file. When a file is deleted, its data blocks are released; for the file system to reuse them, one of the following must be true:

- The file deleted must be the last file stored in the file system, or
- The system must detect the need for compaction and perform it.

---

### Compaction

Compaction is a way of recovering data blocks by shifting files until the gaps left behind by deleted files are eliminated. This operation can be very expensive, but it is necessary because of the method used by bfs to store and delete files.

The system recognizes the need for compaction and performs it when:

- The system has reached the end of the file system and there are still free blocks available, or
- The system deletes a very large file and the file after it on disk is small and is the last file in the file system. (Small files are files of no more than 10 blocks; large files are files of 500 or more blocks.)



## The cdfs File System Type

The cdfs file system is a read-only file system and is used to read CD-ROM disks. CD-ROM stands for Compact Disc – Read Only Memory. The device has a capacity of 600 MB. CD-ROM discs can only be read from, they can never be written to nor can any data be deleted. The CD-ROM discs are primarily used to store documentation such as technical documentation with diagrams and flowcharts, text encyclopedias and dictionaries with graphics, geographic maps, statistical data with graphical output, etc.

NCR UNIX SVR4 uses CD-ROM to store the reference manuals and guides. In other words, all the documentation in the books can now be made available online. This means that all the charts and graphical illustrations contained in any of the manuals can be made available online. The NCR-UNIX SVR4 on-line documentation set also makes use of what is referred to as *hyperlinks*. For example, if you look up a command which makes a reference to another command, you can select to automatically have the manual page for the referenced command extracted. In addition, new or updated versions of the documentation books can be distributed much faster when they are on CD-ROM discs than when they are hard bound books.

In order for a CD-ROM to be read by computer operating systems it must also be formatted. UNIX SVR4 supports the *High Sierra and ISO 9660* format. The *ISO 9660* format is approved by the National Information Standards Organization (NISO) and the International Standards Organization (ISO).

CD-ROM discs all use the same physical format of the disc. CD-ROM discs have addressable physical blocks of 2K byte, also called logical sectors. Logical blocks can be 512, 1024, 2048 bytes long. Both logical sectors and logical blocks start at 1, and are continuous.

A CD-ROM is read in random access. This makes reading from a CD-ROM faster than reading from a streaming tape which reads sequentially. On the other hand, reading from a CD-ROM disk is much slower than reading from a hard disk. Compared to a floppy disk, a CD-ROM device can read data faster. However, the real advantage of a CD-ROM device over a floppy disk is that it stores several hundred megabytes of data compared to the one and a half megabyte of storage for a floppy disk.

A CD-ROM disk can be mounted like any other disk or flex disk device. For example:

```
mount -F cdfs -o1,m /dev/dsk/c0t5d0s0 /cdrom
```

Above command mounts the CD-ROM device with SCSI ID 5 on the directory */cdrom* using the file system specific options.

Option *-l* maps all uppercase filename characters to lowercase

Option *-m* suppresses the version number of the file. Refer to the manual pages for *mount(1M)* and *mount\_cdfs(1M)* for more information.

The file names on a CD-ROM device appear in uppercase letters and often have a version number affixed to them, for example *FILENAME.SUFFIX;1*. Some applications reading from a CD-ROM device may require you to use specific options when mounting a CD-ROM device to convert the uppercase letters to lowercase letters and to truncate the version number of the file.

## **Device Names and Numbers**

**This appendix contains information about device names and numbers.**

|                                             |            |
|---------------------------------------------|------------|
| <b>Major and Minor Device Numbers .....</b> | <b>B-2</b> |
| <b>Device Names and Minor Numbers .....</b> | <b>B-5</b> |

## Major and Minor Device Numbers

The major numbers for Release 2.00 are assigned as follows:

- Major numbers for devices that are part of the BASE operating system are defined as shown in Figure B-1.
- Major numbers for devices that are added by the addition of a package (**pkgadd**) are coded as 0 and are dynamically assigned when the package is installed. Therefore, the major numbers vary from one installation of the package to another.

For both BASE and **pkgadd** devices, the minor numbers are assigned and the corresponding files for device nodes created when you install the BASE O.S. or the package.

Figure B-1, Figure B-2, and Figure B-3 indicate the major number, the minor number, and the name of the corresponding file in the `/dev` directory for devices that are part of the BASE operating system.

| Major # | Minor # | Description                                                          |
|---------|---------|----------------------------------------------------------------------|
| 1       | 132     | Flex disk drive 0:<br><code>/dev/rdisk/f0</code> (default density)   |
|         | 128     | <code>/dev/rdisk/f0t</code> (default density, total disk)            |
|         | 100     | <code>/dev/rdisk/f03d</code> (3.5" flex, double density)             |
|         | 96      | <code>/dev/rdisk/f03dt</code> 3.5" flex, double density, total disk) |
|         | 116     | <code>/dev/rdisk/f03h</code> (3.5" flex, high density)               |
|         | 112     | <code>/dev/rdisk/f03ht</code> (3.5" flex, high density, total disk)  |
| 1       | 133     | Flex disk drive 1:<br><code>/dev/rdisk/f1</code> (default density)   |
|         | 129     | <code>/dev/rdisk/f1t</code> (default density, total disk)            |
|         | 101     | <code>/dev/rdisk/f13d</code> (3.5" flex, double density)             |
|         | 97      | <code>/dev/rdisk/f13dt</code> 3.5" flex, double density, total disk) |
|         | 117     | <code>/dev/rdisk/f13h</code> (3.5" flex, high density)               |
|         | 113     | <code>/dev/rdisk/f13ht</code> (3.5" flex, high density, total disk)  |

Figure B-1: Major and Minor Numbers in BASE O.S. (1 of 3)

| Major # | Minor # | Description                                                                                                                                                                                  |
|---------|---------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 2       | 0       | Pseudo devices:                                                                                                                                                                              |
|         | 1       | <code>/dev/mem</code> (physical main memory)                                                                                                                                                 |
|         | 2       | <code>/dev/kmem</code> (kernel virtual memory and I/O)                                                                                                                                       |
|         | 3       | <code>/dev/null</code> (EOF and bit bucket)                                                                                                                                                  |
|         | 4       | <code>/dev/pmem</code> (any physical memory)                                                                                                                                                 |
| 3       | 0       | <code>/dev/zero</code> (zero fill)                                                                                                                                                           |
|         | 0       | Serial TTY devices connected to rear panel (using software flow control):<br><code>/dev/tty00</code> , <code>/dev/tty00s</code> , <code>/dev/term/00</code> , and <code>/dev/term/00s</code> |
|         | 128     | Serial TTY devices connected to rear panel (using hardware flow control):<br><code>/dev/tty00h</code> and <code>/dev/term/00h</code>                                                         |
|         | 5       | VGA Console: <code>/dev/console</code> , <code>/dev/syscon</code> , <code>/dev/systty</code> , and <code>/dev/ut00</code>                                                                    |
|         | 1 – 8   | Virtual terminals: <code>/dev/vt01</code> through <code>/dev/vt08</code>                                                                                                                     |
| 7       | 0 – 2   | Parallel printers: <code>/dev/lp</code> through <code>/dev/lp2</code>                                                                                                                        |
| 8       | 0       | <code>/dev/clock</code> and <code>/dev/rtc</code>                                                                                                                                            |
| 9       | 5       | STREAMS error log device: <code>/dev/log</code>                                                                                                                                              |
| 13      | 0 – 7   | <code>/dev/xto000</code> through <code>/dev/xto007</code> and <code>/dev/xto/000</code> through <code>/dev/xto/007</code>                                                                    |
| 14      | 0 – 57  | <code>/dev/sxto000</code> through <code>/dev/sxto071</code> and <code>/dev/sxto/000</code> through <code>/dev/sxto/071</code>                                                                |
| 16      | 0       | Generic TTY (indirect driver for controlling TTY device: <code>/dev/tty</code> )                                                                                                             |
| 17      | 0       | <code>/dev/osm</code>                                                                                                                                                                        |
| 18      | 0, 1    | CMOS RAM driver: <code>/dev/cram</code> , <code>/dev/cramx</code>                                                                                                                            |
| 19      | 0       | VGA Console: <code>/dev/sysmsg</code>                                                                                                                                                        |
| 20      | 0 – 14  | Keyboard option driver: <code>/dev/kd/kdvm00</code> through <code>/dev/kd/kdvm14</code>                                                                                                      |
| 25      | 0, 1    | STREAMS administrative drivers:<br><code>/dev/sad/user</code> and <code>/dev/sad/admin</code>                                                                                                |

Figure B-2: Major and Minor Numbers in BASE O.S. (2 of 3)

## Device Names and Numbers

| Major # | Minor #         | Description                                                                                                                             |
|---------|-----------------|-----------------------------------------------------------------------------------------------------------------------------------------|
| 29      | 0               | Video controller option driver: <i>/dev/video</i>                                                                                       |
| 29      | 1               | Driver for video device: <i>/dev/vidadm</i>                                                                                             |
| 30      | 0 – 14          | Keyboard driver: <i>/dev/kd/kd00</i> through <i>/dev/kd/kd14</i>                                                                        |
| 30      | 15              | <i>/dev/vtmon</i>                                                                                                                       |
| 33      | 0 – 7           | <i>/dev/xt000</i> through <i>/dev/xt007</i> and <i>/dev/xt/000</i> through <i>/dev/xt/007</i>                                           |
| 34      | 0 – 57          | <i>/dev/sxt000</i> through <i>/dev/sxt071</i> and <i>/dev/sxt/000</i> through <i>/dev/sxt/071</i>                                       |
| 35      | 0 – <i>xxxx</i> | SCSI devices such as hard disks and cartridge tapes (Additional information about the minor numbers is provided later in this chapter.) |

Figure B–3: Major and Minor Numbers in BASE O.S. (3 of 3)

For more information about device names and numbers, see the “Device Names and Minor Numbers” section of this appendix.

## Device Names and Minor Numbers

When you install the BASE operating system, device names and numbers are created for all devices that are physically present on your system as well as for certain other common devices shown earlier in Figure B-1, Figure B-2, and Figure B-3.

The device name and the 22 bit minor number for SCSI devices are determined by the location and characteristics of the device in the system. This involves, for example, not only the SCSI ID, LUN, and slicing or tape control, but also the SCSI bus, the SCSI controller, and the system bus. Note that in Release 2.01, there are now two system busses for the system 3550 ( the primary and secondary Micro Channel ).

Since different models of the System 3000 have different configurations and even the same models can be configured differently, the device names and numbers vary from machine to machine. This section provides examples of some possible device names and numbers. The last section in this appendix shows how to calculate the SCSI device name and number for a device according to its location in the system.

For any device that exists on your system, you can use the `/usr/sadm/sysadm/bin/nodes` command to determine the device name if you know the minor number or to determine the major and minor numbers if you know the device name.

- To determine the device name for a device, enter:

```
/usr/sadm/sysadm/bin/nodes -m minor_number
```

- To determine the minor number for a device, enter:

```
/usr/sadm/sysadm/bin/nodes -d full_device_pathname
```

---

## SCSI Hard Disks

**NOTE:** The default factory installation for the System 3000 Model 3550 attaches the backplane for the fixed media to the second SCSI bus (bus 1) on the first SCSI controller (controller 0) on the primary Micro Channel (system bus 0).

**/dev/(r)dsk/c1t0d0s0**

First system bus, first SCSI controller, second SCSI bus, SCSI device id=0, LUN=0, slice 0 (major number is 35, minor number is 16384)

**/dev/(r)dsk/c1t0d0s1**

First system bus, first SCSI controller, second SCSI bus, SCSI device id=0, LUN=0, slice 1 (major number is 35, minor number is 16385)

**/dev/(r)dsk/c1t6d0s0**

First system bus, first SCSI controller, second SCSI bus, SCSI device id=6, LUN=0, slice 0 (major number is 35, minor number is 16480)

**/dev/(r)dsk/c1t6d0s1**

First system bus, first SCSI controller, second SCSI bus, SCSI device id=6, LUN=0, slice 1 (major number is 35, minor number is 16481)

**/dev/(r)dsk/c100t0d0s0 (system 3550)**

Second system bus, first SCSI controller, first SCSI bus, SCSI device id=0, LUN=0, slice 0 (major number is 35, minor number is 524288)

**/dev/(r)dsk/c100t0d0s1 (system 3550)**

Second system bus, first SCSI controller, first SCSI bus, SCSI device id=0, LUN=0, slice 1 (major number is 35, minor number is 524289)



## SCSI Cartridge Tape

**NOTE:** The default factory installation for the System 3000 Model 3550 attaches the backplane for the removable media to the first SCSI bus (bus 0) on the first SCSI controller (controller 0) on the primary Micro Channel (system bus 0).

### **/dev/rmt/c0t0d0s0**

First system bus, first SCSI controller, first SCSI bus, SCSI device id=0, LUN=0, rewind on open and close (major number is 35, minor number is 3)

### **/dev/rmt/c0t0d0s0ny**

Same as above, except no rewind on open but rewind on close (major number is 35, minor number is 2)

### **/dev/rmt/c0t0d0s0yn**

Same as above, except rewind on open but no rewind on close (major number is 35, minor number is 1)

### **/dev/rmt/c0t0d0s0n and /dev/(r)dsk/c0t0d0s0nn**

Same as above, except no rewind on open or close (major number is 35, minor number is 0)

### **/dev/rmt/c0t3d0s0**

First system bus, first SCSI controller, first SCSI bus, SCSI device id=3, LUN=0, rewind on open and close (major number is 35, minor number is 51)

### **/dev/rmt/c0t3d0s0ny**

Same as above except no rewind on open but rewind on close (major number is 35, minor number is 50)

### **/dev/rmt/c0t3d0s0yn**

Same as above except rewind on open but no rewind on close (major number is 35, minor number is 49)

### **/dev/rmt/c0t3d0s0n and /dev/rmt/c0t3d0s0nn**

Same as above except no rewind on open & close (major number is 35, minor number is 48)

## Minor Numbers for SCSI Devices

The minor numbers for all devices are determined using 22 bits. Figure B-4 shows the minor number bit definitions for SCSI devices in Release 2.01.

| Component                  | # of Bits |                                                            |
|----------------------------|-----------|------------------------------------------------------------|
| I/O or System Bus          | 3         | Normally 0. The secondary Micro Channel in a 3550 is 1.    |
| SCSI Controller            | 3         | The first (or only) controller is 0.*                      |
| SCSI Bus                   | 2         | The first (or only) bus is 0.                              |
| Logical Unit Number (LUN)  | 6         | Frequently 0 (except for RAID). (High order bit reserved.) |
| Physical Unit Number (PUN) | 4         | Same as the SCSI ID.                                       |
| Slice                      | 4         | Slice number for hard disks.                               |
| * See NOTE.                |           |                                                            |

| I/O or<br>System<br>Bus | SCSI<br>Controller | SCSI Reser-<br>ved | LUN   | PUN       | Slice   |
|-------------------------|--------------------|--------------------|-------|-----------|---------|
| 0 0 0                   | 0 0 0              | 0 0                | 0     | 0 0 0 0 0 | 0 0 0 0 |
| 3 bits                  | 3 bits             | 2 bits             | 1 bit | 5 bits    | 4 bits  |

Figure B-4: Minor Number Bit Definitions

**NOTE:** The SCSI Controller is determined by the logical ordering of the "IO Space" parameter of the SCSI adapter setup. For example, suppose you have adapters with IO Space values of 400h, 800h, c00h, and 1200h. No matter which physical order the

adapters are in, the adapter with IO Space 400h is controller 0, the adapter with IO Space 800h is controller 1, the adapter with IO Space c00h is controller 2, and the adapter with IO Space 1200h is controller 3.

The SCSI device name is written `cABCtXdYsZ`. If `ABC` contains leading zeros, they are dropped (except that at least one digit must follow the `c`, even if the digit is 0). For example, `c000t1d0s1` is written `c0t1d0s1` and `c010t1d0s1` is written `c10t1d0s1`. Figure B-5 shows how the SCSI device name is composed.

| System Component           | # of Bits |          | Contribution to SCSI Device Name |
|----------------------------|-----------|----------|----------------------------------|
| I/O or System Bus          | 3         | <b>A</b> | <code>cABCtXdYsZ</code>          |
| SCSI Controller            | 3         | <b>B</b> | <code>cABCtXdYsZ</code>          |
| SCSI Bus                   | 2         | <b>C</b> | <code>cABCtXdYsZ</code>          |
| Logical Unit Number (LUN)  | 5         | <b>Y</b> | <code>cABCtXdYsZ</code>          |
| Physical Unit Number (PUN) | 4         | <b>X</b> | <code>cABCtXdYsZ</code>          |
| Slice                      | 4         | <b>Z</b> | <code>cABCtXdYsZ</code>          |

Figure B-5: SCSI Device Name Components

**NOTE:** The LUN and PUN do not appear in the same order in the device name as they do in the bit pattern. LUN comes to the left of PUN in the bit pattern, but LUN comes to the right of PUN in the device name.

### Examples:

Figure B–6 shows the bits set for slice 0 on the hard disk with SCSI ID = 0 that uses the second SCSI bus on the first SCSI controller on the first system bus. The device name is */dev/rdisk/c1t0d0s0* and the minor number is 16384.

| I/O or<br>System<br>Bus | SCSI<br>Controller | SCSI<br>Bus | Reserved | LUN       | PUN     | Slice   |
|-------------------------|--------------------|-------------|----------|-----------|---------|---------|
| 0 0 0                   | 0 0 0              | 0 1         | 0        | 0 0 0 0 0 | 0 0 0 0 | 0 0 0 0 |
| 3 bits                  | 3 bits             | 2 bits      | 1 bit    | 5 bits    | 4 bits  | 4 bits  |

Figure B–6: Example 1 of SCSI Device Name and Number

Figure B–7 shows the bits set for slice 1 on the hard disk with SCSI ID = 1 that uses the second SCSI bus on the second SCSI controller on the first system bus. The device name is */dev/rdisk/c1t1d0s1* and the minor number is 81937.

| System<br>Bus | SCSI<br>Controller | SCSI<br>Bus | Reserved | LUN       | PUN     | Slice   |
|---------------|--------------------|-------------|----------|-----------|---------|---------|
| 0 0 0         | 0 0 1              | 0 1         | 0        | 0 0 0 0 0 | 0 0 0 1 | 0 0 0 1 |
| 3 bits        | 3 bits             | 2 bits      | 1 bit    | 5 bits    | 4 bits  | 4 bits  |

Figure B–7: Example 2 of SCSI Device Name and Number

---

## Flex Disks

### **/dev/rdisk/f03ht**

Drive 0, 3.5", high density, double sided (total disk) (major number is 1, minor number is 112)

### **/dev/rdisk/f13ht**

Drive 1, 3.5", high density, double sided (total disk) (major number is 1, minor number is 113)

You should format a flex disk using the device name that matches the density of the flex disk. Then, you can use utilities that access the device name without the density (for example, **/dev/rdisk/f0t**) to read or write to the flex. The device driver recognizes the size (density) of a flex disk only after the flex disk has been formatted.

---

## Serial I/O Ports (TTY)

### **/dev/tty00s**

Software control using XON/XOFF (major number is 3, minor number is 0)

### **/dev/tty00h**

Hardware control using DTR (major number is 3, minor number is 128)

---

## Console (VGA Monitor)

### **/dev/console**

VGA Monitor; also first virtual terminal – **/dev/vt00** (major number is 5, minor number is 0)

---

## 8– or 16–Port Serial Controller Boards

**/dev/acl/00**

First board

**/dev/acl/01**

Second board

The major number is dynamically allocated/assigned when you install the **tty** driver/package. Therefore, the major number may be different on every system, depending on which drivers/packages are linked into the kernel before the **tty** package is installed.

The minor number is the same as the number of the board, such as **0** for the first 8– or 16–Port Serial Controller board.

---

## Serial (TTY) I/O Ports

**/dev/term/s00**

First port (P1) on first 8– or 16–Port Serial Controller board.

**/dev/term/s08**

First port (P1) on second 8–Port Serial Controller board OR ninth port (P9) on the first 16–Port Serial Controller board.

**/dev/term/s31**

Last port (P8) on fourth 8–Port Serial Controller board OR last port (P16) on second 16–Port Serial Controller board.

The major number is dynamically allocated/assigned when you install the **tty** driver/package. Therefore, the major number may be different on every system, depending on which drivers/packages are linked into the kernel before the **tty** package is installed.

The TTY ports are numbered sequentially, beginning with the first port on the distribution box attached to the serial controller board with lowest I/O address. The minor number is the same as the device name, such as **00** for */dev/term/s00* (port 1 on the first distribution box connected to the first 8- or 16-Port Serial Controller board) or **31** for */dev/term/s31* (the last port on either the fourth distribution box connected to the first 8-Port Serial Controller board or the second distribution box connected to the second 16-Port Controller board).

---

## Pseudo Devices

### */dev/mem*

Physical (main) memory (major number is 2, minor number is 0)

### */dev/kmem*

Logical (kernel) memory (major number is 2, minor number is 1)

### */dev/null*

EOF and bit bucket (major number is 2, minor number is 2)

### */dev/pmem*

Physical memory (major number is 2, minor number is 3)

### */dev/zero*

Zero fill (major number is 2, minor number is 4)

**NOTE:** The */dev/zero* device is similar to the */dev/null* device except a read from */dev/zero* fills the user's buffer with zeroes. If */dev/zero* is mapped, it provides a scratch area (zero filled) for that process.

---

## Virtual Terminals

### */dev/vt01*

(major number is 5, minor number is 1)

### */dev/vt08*

(major number is 5, minor number is 8)

---

**Parallel Printer  
Port**

**/dev/lp**

(major number is 7, minor number is 0)



UNIX System V Release 4 provides a menu interface to the most common administrative procedures. It is invoked by executing the **sysadm** command and so is referred to as the **sysadm** interface.

Additions or changes can be made directly to this interface by using two shell commands, **edsysadm** and **delsysadm**, or they can be delivered as part of a software package. This appendix provides instructions for making direct changes to the interface using **edsysadm** and **delsysadm**. Instructions for delivering changes in a package are covered in the *UNIX SVR4 Programmer's Guide: System Services and Application Packaging Tools*.

|                                                 |      |
|-------------------------------------------------|------|
| Interface Structure: a Hierarchy of Menus ..... | C-2  |
| Writing Your Help Messages .....                | C-8  |
| Creating or Changing a Menu Entry .....         | C-18 |
| Creating or Changing a Task Entry .....         | C-25 |
| Deleting a Menu or Task Entry .....             | C-31 |

## Interface Structure: a Hierarchy of Menus

The sysadm interface consists of a hierarchy of menus. At the top of the hierarchy is the main menu (labeled UNIX System V Administration). It appears on your screen, immediately after you invoke sysadm, as shown in Figure C-1:

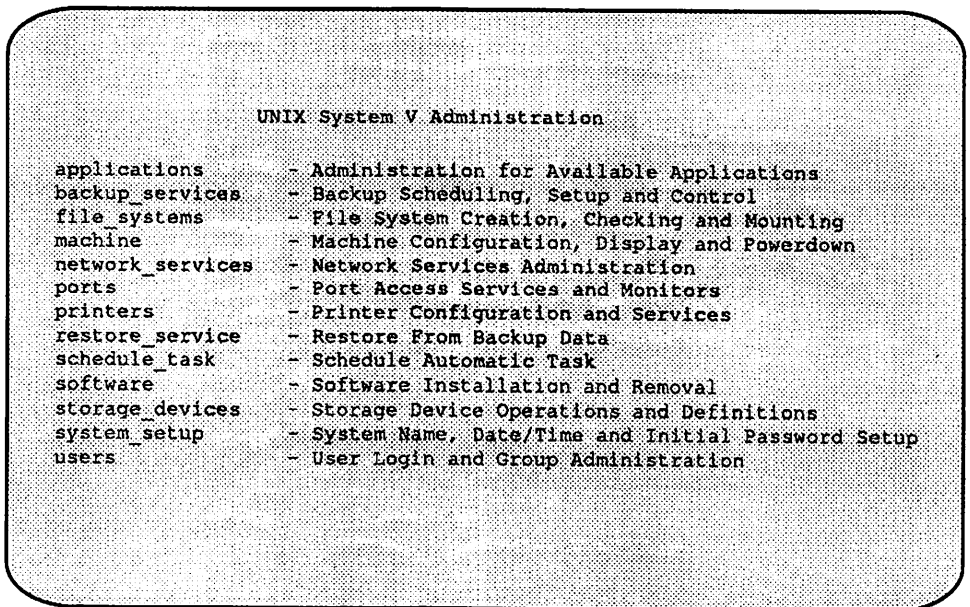


Figure C-1: Sysadm Main Menu

**NOTE:** The applications menu does not appear on the main menu until at least one menu or task has been placed under it.

## Menus and Tasks

The main menu consists of a list of function specific menus. The lefthand column notes the menu names (such as machine) and the righthand column gives descriptions of these menus. Each menu offers other menus and/or names of tasks. For example, the machine menu, shown in Figure C-2, contains one menu (configuration) and four tasks.

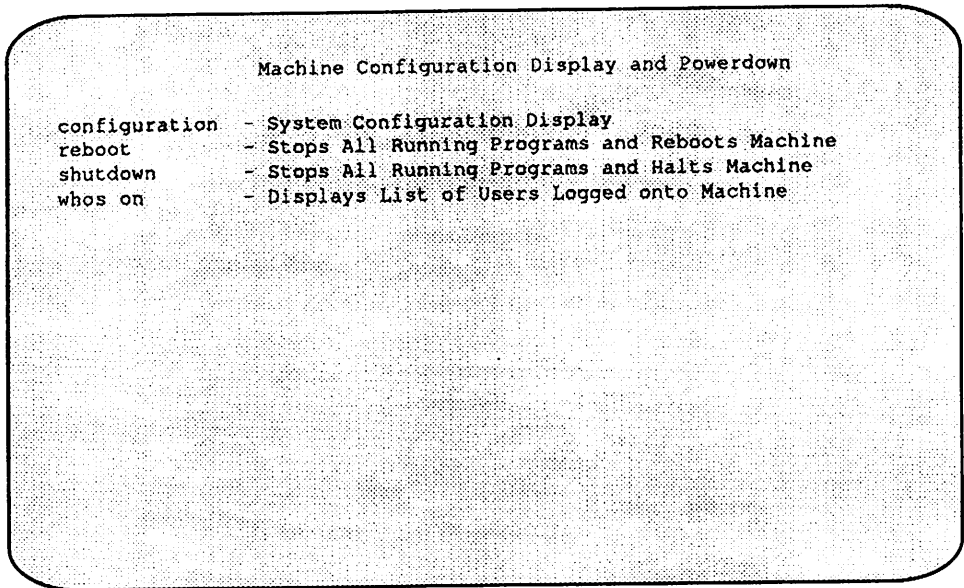


Figure C-2: Sysadm Machine Menu

Choosing the menu entry from this screen causes another menu to be presented. Choosing a task entry begins execution of that task.

---

## Selecting a Name and Location for an Interface Entry

Before making a change to the interface, (either by modifying an existing menu or task or by adding a new one) you need to know the name, description, and location of the menu or task being added or changed, as defined below.

### Name

The name of the menu or task as it will appear in the lefthand column of the screen.

### Description

The description of the menu or task as it will appear in the righthand column on the screen.

### Location

The location of menu or task in the sysadm menu hierarchy. This location is a combination, step-by-step, of all the menu names that must be chosen to reach the menu or task. Each step must already exist when the entry is added. For example, when you add a task with a location of **main:applications:mypkg**, you must already have created an entry for the menu **mypkg**.

All locations begin with **main**. When defining a location in the procedures that follow, each step should be separated by a colon. For example, the **powerdown** task is under the menu **machine**, which, in turn, is under the **main** menu. Thus, the location of the **powerdown** task is **main:machine:powerdown**.

You can create new **sysadm** menus at any level and you can change or add to any of the original **sysadm** menus. You should be aware, however, that if you make changes to original menus you might cause problems in the execution of standard **sysadm** operations. It is therefore recommended (though not mandatory) that you create new menus for your interface changes. The main **sysadm** menu includes a menu entry called **applications** that is actually empty when your system is delivered. It is recommended (though not mandatory) that you use this menu to store any new menus and task that are associated with software application packages.

---

## The edsysadm Command

The **edsysadm** command, which allows you to make changes or additions to the interface, is an interactive command that functions much like the **sysadm** command itself. It presents a series of prompts for information. (Which prompts appear depends on your response to them.)

After you have responded to all the prompts, **edsysadm** presents a form that you must fill in with information describing the menu or task being changed or added. This form is called the menu (or task) definition form. If you are changing an existing menu or task entry, the definition form is already filled in with the current values, which you can edit. If you are adding a new menu or task entry, the form is empty and you must fill it in.

This appendix describes the prompts presented by **edsysadm**, describes your choice of responses for each prompt, and tells you which response to choose for a particular course of action. Illustrations showing how these prompts appear on the screen are not included.

### Item Help Files for Menus and Tasks

For every menu or task that you add with the **edsysadm** command, you must also provide an item help file. The messages in this file are shown on a user's screen when that user requests help while on a menu screen or within a form. You can create an item help file with any editor but it must follow the format described in the "Writing Your Help Messages" section.

Because an item help file is required for every menu and task, you cannot skip the field for the name of the file when filling out the menu or task definition form. If you fill in this field with a filename that cannot be found, you are placed in editor and can create an item help file at that time. **edsysadm** creates this new file in your current working directory and names it *Help*.

## Executable Files for Tasks

Before adding a task to a menu, you must already have the executable files that will be run when a user chooses that task. You may write these yourself or a programmer may supply them to you. In either case, this appendix covers only the procedures used to make the changes to the interface. Instructions for writing the executables can be found in the *UNIX SVR4 Programmer's Guide System Services and Application Packaging Tools*.

---

### The `delsysadm` Command

The `delsysadm` command allows you to delete a task or menu from the interface. It checks for dependencies on the entry being removed before it deletes it. (A dependency exists if the menu being removed contains an entry placed there by an application package.) If a dependency is found, you are asked whether you want to continue with the removal.

When you delete a menu entry, it must already be empty (contain no other menus or tasks). To remove a menu and all its entries at the same time, you can execute `delsysadm` with the `-r` option.

### Caution

Use `delsysadm` to remove only those menu or task entries that you have added to the interface with `edsysadm`.

## Writing Your Help Messages

You must write help messages for every interface modification you make. They must be in what is called an item help file. This file has text for two types of messages:

- The help message that is shown when the user requests help from the parent menu
- The help message that is shown for each field when your task action is a FACE (Framed Access Command Environment) form

The format of the item help file allows you to create one time help file for each task, combine all of your help messages for multiple tasks into one file, use the same message for multiple FACE forms, and to define a title hierarchy for the help message screens.

---

### The Item Help File

There are no naming restrictions for the item help file when it resides on your machine. However, when it is placed into the interface structure, the item help file must always be named *Help*. Since **edsysadm** creates the directory structure required by the interface, it gives the file you name on the menu (or task) definition the appropriate location and correct name regardless of its name on your machine. This means that you do not have to name your item help files *Help* and therefore, can have more than one item help file in your working directory at the same time. **edsysadm** handles the details of giving it the correct name.



There are three types of entries in an item help file:

- The menu item help
- The default title (can define both a global default and a form default)
- The field item help

A description of each type of entry and its format follows. All of the entries use the colon (:) as the keyword delimiter

### The Menu Item Help Message Format

The menu item help is shown whenever a user requests help while viewing the menu screen and the cursor is on the task or menu item. Menu item help must be written for both menu and task entries. For example, if you add a menu under main:applications and that menu has three tasks under it, you need to deliver four menu item help messages.

The format for the menu item help definition is as follows:

```
{task_name:}ABSTRACT:
<TAB> Line 1 of message text
<TAB> Line 2 of message text
<TAB> Line n of message text
```

*task\_name* defines the task (or menu) entry to which this help message belongs. This name must match the name that you have decided should appear in the lefthand column of the menu screen. *task\_name* is not optional when more than one menu item help definition is defined in the same item help file. This helps in identifying the task or menu to which the message belongs.

The message text should be entered beneath the header line. There can be multiple lines of text with a maximum length of 69 characters per line. Each line must begin with a tab character. Blank lines may be included within the message as long as they also begin with a tab character. An example menu item help definition is shown below.

```
task1:ABSTRACT:
 This is line one of the menu item help message.
 This is a second line of message text.

 The preceding line will appear as a blank line
 when the help message is shown because it begins
 with a tab.
```

The title for a menu item help message is always the description text, as it appears in the lefthand column of the menu display, prepended by the string “Help on.”

### The Default Title Format

You can define two types of default titles:

- A global default title to be used on all of the help messages defined in the item help file
- A form default title to be used on all of the help messages defined for a particular form in an item help file with messages defined for numerous FACE forms (delivered as task actions)

Defaults can be overridden, as described in the section named “The Title Hierarchy.”

The format for the default title definition is as follows:

```
[form_id]TITLE:Title Text
```

*form\_id* is the name of the form as it is defined with `lininfo` in your FACE form definition. When a *form\_id* is supplied, this line defines a form default title. When it is not supplied, this line defines a global default title.

The title text defined after the TITLE keyword has the string HELP on prepended to it when displayed. Keep this in mind when writing the title.

An example form default title definition is shown below

```
task1:TITLE:Package Administration Task1
```

If task1 had not been added before TITLE, this example would be defining a global default title. The title defined by the example above is displayed as:

```
HELP on Package Administration Task1
```

## The Field Item Help Message Format

The field item help message is shown whenever a user requests help from within a FACE form. Each field on the form must have a help message defined in the item file.

The format for the field item help definition is as follows:

```
{form_id:]field_id:[Title Text]
<TAB> Line 1 of message text
<TAB> Line 2 of message text
<TAB> Line n of message text
```

*form\_id* is the name of the form as it is defined with lininfo in your FACE form definition. When one item help file contains messages for multiple tasks (and so multiple forms), it is used to distinguish with which form a field belongs. It is optional if the file contains message for only one task. *field\_id* is the name of the field as it is defined with lininfo in your FACE form definition. *Title text* defines a title used only with the help message for this field. As with the default title, the text defined here has the string “HELP on” prepended to it when displayed.

The message text should be entered beneath the header line. There can be multiple lines of text with a maximum length of 69 characters per line. Each line must begin with a tab character. Blank lines may be included within the message as long as they also begin with a tab character. An example field item help definition is shown below.

```
task1:fld1:the Name Field
 This is the text for a field item help for a name
 field.

 The preceding line will appear as a blank line
 when the help message is shown because it begins
 with a tab.
```

The title for this field item help message, as defined above, would be “HELP on the Name Field.”

---

### The Title Hierarchy

You can define a global default title, a form default title, and a field title in an item help file. When all three are defined in the same file, the following rules are followed:

- The global default title is used for any message defined in an item help file that does not have a form default title or field title.
- The form default title is used for any message defined in an item help file and that is associated with the form, unless it has a field title.
- The field title is used only for the one help message for which it is defined.

In summary, if no field title is defined, the form default title is used. If no form default title is defined, the global default title is used. You always want at least a global default title defined; otherwise, the string “HELP on” is displayed with no descriptive text.

To define a global default title, add a line to your item help file in the following format:

```
TITLE:Title Text
```

where *Title Text* is the text for the global default title.

To define a form default title, add a line to your item help file in the following format:

```
form_id:TITLE:Title Text
```

where *form\_id* is the name of form as it is defined with lininfo in your FACE form definition and *Title Text* is the text for the form default title.

To define a field title, use the following format for the field item help header line:

```
form_id:field_id:Title Text
```

where *form\_id* is the name of the form as it is defined with lininfo in your FACE form definition, *field\_id* is the name of the field as it is defined with lininfo in your FACE form definition and *Title Text* is the text for the field title.

**NOTE:** In all cases, the text defined as *Title Text* is always prepended with the string “HELP on” when displayed to a user

---

## Setting Up for Item Help in a FACE Object

To help the interface read your item help file and know with which forms and fields a help message is associated, the help and lininfo descriptors in the FACE object definition must be defined as follows:

- The help descriptor must be defined exactly as shown on the line below:

```
help=OPEN TEXT $INTFBASE/Text.itemhelp $LININFO
```

- The lininfo descriptor for each field must be defined as

```
lininfo={form_id:}field_id
```

where *form\_id* and *field\_id* are names each no longer than 30 characters. The names defined here as *form\_id* and *field\_id* must match exactly those used as *form\_id* and *field\_id* in the item help file.

**NOTE:** In all cases, HELP precedes the text defined as *Title Text* when display to a user.

---

## Example Item Help Files

This section shows two example item help files. Figure C-3 shows an item help file that defines messages for only one FACE form; Figure C-4 shows its associated FACE form. Figure C-5 shows an example of defining messages for multiple FACE forms in one item help file; Figure C-6 shows its associated FACE form.

**ABSTRACT:**

The text defined here will be shown to users when they request help while viewing the parent menu for this task. The task name is "adding users."

**TITLE:Adding Users****field1:**

The text defined here will be shown to users when they request help from the form and the cursor is positioned at field1. The title for this message will be "HELP on Adding Users" as defined above.

**field2:Field 2**

The text defined here will be shown to users when they request help from the form and the cursor is positioned at field2. The title for this message will be "HELP on Field 2".

Figure C-3: Item Help File for One Form

*Note: The lininfo descriptors in the form definition associated with this file should look like this:*

```
lininfo=field1
```

```
lininfo=field2
```

Figure C-4: FACE Form Associated with Item Help File in Figure C-3

```
add:ABSTRACT:
 The text defined here will be shown to users when they
 request help while viewing the parent menu for the task
 named add.

add_user:TITLE:Adding Users

add_user:field1:
 The text defined here will be shown to users when they
 request help from the form and the cursor is positioned
 at field1. The title for this message will be "HELP on
 Adding Users" as defined above.

add_user:field2:Field 2
 The text defined here will be shown to users when they
 request help from the form and the cursor is positioned
 at field2. The title for this message will be "HELP on
 Field 2".

delete:ABSTRACT:
 The text defined here will be shown the users when they
 request help while view the parent menu for the task
 named delete.

delete_user:TITLE:Deleting Users

delete_user:field1:
 The text defined here will be shown to users when
 request help from the form and cursor is positioned at
 field1. The title for this message will be "HELP on
 Deleting Users" as defined above.

delete_user:field2:Field 2
 The text defined here will be shown to users when they
 request help from the form and the cursor is positioned
 at field2. The title for this message will be "HELP on
 Field 2".
```

Figure C-5: Item Help File for Multiple Forms



*Note: The lininfo descriptors in the form definition associated with this file should look like this:*

```
.
.
lininfo=add_user:field1
.
.
lininfo=add_user:field2
.
.
lininfo=delete_user:field1
.
.
lininfo=delete_user:field2
.
.
```

Figure C-6: Item Help File for Multiple Forms

## Creating or Changing a Menu Entry

The procedures for creating a new menu and for changing an existing one are similar and both result in the display of a menu definition form. Each procedure is described below, followed by a description of the menu definition form.

---

### Creating a Menu Entry

Before creating a new menu entry, you should:

- Select a name and description for the menu
- Select a location for it in the interface
- Prepare an item help file for the menu entry (refer to the “Writing Your Help Messages” section earlier in this appendix for instructions)

Step 1: Type **edsysadm** and press ENTER.

**NOTE:** If you do not execute this command from the directory in which the item help file resides, supply the full pathname when prompted for the name of the item help file.

Step 2: You are asked to choose between a menu and a task. Choose **menu** and ENTER.

Step 3: You are asked to choose between adding a new menu and changing an existing one. Choose **add** and press ENTER.

- Step 4:** You are given an empty menu definition form. Fill it in and press **SAVE**. (See “The Menu Definition Form” for descriptions of the fields on this form.)
- Step 5:** You are asked if you want to test the changes before actually making them. Answer **yes** or **no** and press **SAVE**. (If you answer yes, refer to the “Testing Your Menu Changes On-Line” section to learn what the test involves.)
- Step 6:** You are asked if you want to install the modifications into the interface on your machine or save them for a package. Choose **install** and press **SAVE**.
- Step 7:** You see one of the following:
- A confirmation screen if the installation was successful. This means you now have an empty menu. (To populate the menu entry you just created, follow the procedure for adding a new menu entry for adding a new task entry.)
  - An error message describing the reason installation was unsuccessful.
  - A message explaining that there is a conflict with the name you have chosen for this menu. You are prompted to choose from three possible actions: **install** (the addition is installed despite the collision of names), **rename** or **relocate** (you are shown the menu form at which time you can change the name or location of the entry), and **do not install**.

---

## Changing a Menu Entry

Before changing a menu entry, you should:

- Know the name and description for the menu
- Know its location for it in the interface
- Change the associated item help file, if necessary, or create a new one (see the “Writing Your Help Messages” section earlier in this chapter for instructions)

Step 1: Type **edsysadm** and press ENTER.

**NOTE:** If you have changed an item help file or created a new one and you do not execute this command from the directory in which the file resides, supply the full pathname when asked for the name.

Step 2: You are asked to choose between a menu and a task. Choose **menu** and ENTER.

Step 3: You are asked to choose between adding a new menu and changing an existing one. Choose **change** and press ENTER.

Step 4: You are asked if your change is for an on-line menu or for a menu that has been saved for a package. Choose **on-line** and press SAVE.

Step 5: You are asked to supply the name and location of the menu to be changed. Fill in the blanks and press SAVE.

Step 6: You are given a menu definition form filled in with the current values for the menu named above. Make the desired changes and press SAVE. (See “The Menu Definition Form” for descriptions of the fields on this form.)

- Step 7:** You are asked if you want to test the changes before actually making them. Answer yes or no and press **SAVE**. (If you answer yes, refer to the section entitled “Testing Your Menu Changes On-Line” to learn what the test involves.)
- Step 8:** You are asked if you want to install the modifications into the interface on your machine or save them for a package. Choose **install** and press **SAVE**.
- Step 9:** You see one of the following:
- A confirmation screen if the installation was successful. This means you now have an empty menu. (To populate the menu entry you just created, follow the procedure for adding a new menu entry for adding a new task entry.)
  - An error message describing the reason installation was unsuccessful.
  - A message explaining that there is a conflict with the name you have chosen for this menu. You are prompted to choose from three possible actions: install (the addition is installed despite the collision of names), rename or relocate (you are shown the menu form at which time you can change the name or location of the entry), and do not install.

---

## Testing Your Menu Changes On-Line

Before installing your menu changes, you may want to verify that you have added an entry to a menu. The **edsysadm** command gives you a chance to do this after you fill in the menu definition form. Follow these steps to perform your test.

**Step 1:** Type **yes** when **edsysadm** presents the following prompt:

Do you want to test this modification before  
continuing?

**Step 2:** The parent menu (on which your addition or change is listed) is displayed. Check to make sure your modification has been made correctly.

**Step 3:** Put the cursor on the new or changed menu and press the **HELP** key. The text of the help message for that menu entry is displayed so you can check it. (Press **CANCEL** to return to the menu.)

**Step 4:** To exit on-line testing, press the **CANCEL** key.

**Step 5:** You are returned to the prompt:

- If you want to continue executing the change, type **no**.
- If you want to make additional modifications to the menu definition form, press **CANCEL**. You are returned to the form and can make further changes at that time (Press **SAVE** when you have finished your editing. You can then retest your changes or continue executing the change.)

---

## The Menu Definition Form

This form contains four fields in which you must provide the following information: a menu name, a menu description, a menu location, and the name of the help message for the menu. Below are descriptions of the information you must provide in each field.

### Menu Name

The name of the new menu (as it should appear in the lefthand column of the screen). This field has a maximum length of 16 characters and should consist of alphanumeric characters.

### Menu Description

A description of the new menu (as it should appear in the righthand column of the screen). This field has a maximum length of 58 characters and can consist of any alphanumeric character except an at sign (@), circumflex (^), tilde (~), back quote (`), single quote ('), and double quotes (").

### Menu Location

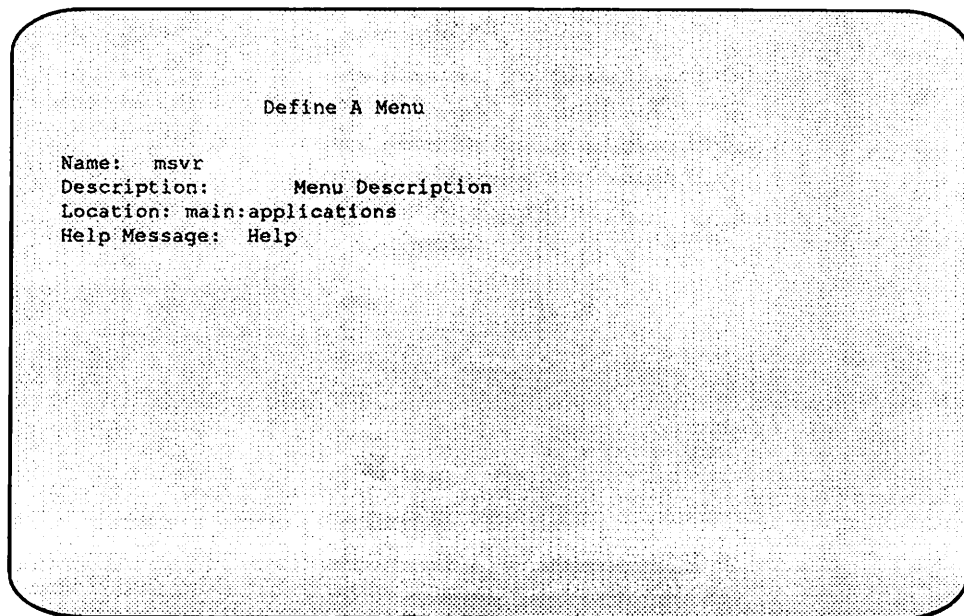
The location of the menu in the menu hierarchy, expressed as a menu pathname. The pathname should begin with the main menu followed by all other menus that must be traversed (in the order they are traversed) to access this menu. Each menu name must be separated by colons. For example, the menu location for a menu entry being added to the Applications menu is `main:applications`. Do not include the menu name in this location definition. The complete path to this menu entry is the menu location plus the menu name defined at the first prompt.

This field has a maximum length of 50 characters and should consist of alphanumeric characters.

### Menu Help File Name

Pathname to the item help file for this menu entry. If it resides in the directory from which you invoked `edsysadm`, you do not need to give a full pathname. If you name an item help file that does not exist, you are placed in an editor (as defined by `$EDITOR`) to create one. The new file is created in the current directory and named *Help*.

Figure C-7 shows a sample menu definition form that has been filled in:



The image shows a sample menu definition form titled "Define A Menu". The form is enclosed in a rounded rectangle with a light gray background. It contains the following text:

```
Define A Menu

Name: msvr
Description: Menu Description
Location: main:applications
Help Message: Help
```

Figure C-7: Sample Menu Definition Form



## Creating or Changing a Task Entry

The procedures for creating a new task and for changing an existing one are similar and both result in the display of a task definition form. Each procedure is described below, followed by a description of a task definition form.

---

### Creating a Task Entry

Before creating a task entry, you should:

- Gather all files that will be associated with this task, such as the help files, FACE forms, and other executables. All files should already be prepared. Instructions for creating these files are provided in the *UNIX SVR4 Programmer's Guide: System Services and Application Packaging Tools*.
- Decide on the task name and description
- Decide on its location in the interface
- Create an item help file (see "Writing Your Help Messages" earlier in the chapter for instructions).

Step 1: Type **edsysadm** and press ENTER.

**NOTE:** If you do not execute this command from the directory in which the files associated with this task reside, enter full pathnames when supplying filenames.

Step 2: You are asked to choose between a menu and a task. Choose **task** and ENTER.

- Step 3: You are asked to choose between adding a new task and changing an existing one. Choose **add** and press **ENTER**.
- Step 4: You are given an empty task definition form. Fill it in and press **SAVE**. (See “The Task Definition Form” for descriptions of the fields on this form. Be aware that, when you name the menu under which you want this new task to reside, that menu must already exist.)
- Step 5: You are asked if you want to install the modifications into the interface on your machine or save them for a package. Choose **install** and press **SAVE**.
- Step 6: You see one of the following:
- A confirmation screen if the installation was successful. This means that the new task is now in the menu and can be executed.
  - An error message describing the reason installation was unsuccessful.
  - A message explaining that there is a conflict with the name you have chosen for this menu. You are prompted to choose from three possible actions: **install** (the addition is installed despite the collision of names), **rename** or **relocate** (you are shown the menu form at which time you can change the name or location of the entry), and **do not install**.

---

## Changing a Task Entry

Before changing a task entry, you should:

- Gather any of the files associated with this task that have been changed or are new. All files should already be prepared or changed. Instructions for creating these files are provided in the *UNIX SVR4 Programmer's Guide: System Services and Application Packaging Tools*.
- Know the menu name and description
- Know its location in the interface
- Change the associated item help file, if necessary (see “Writing Your Help Messages” earlier in this chapter for instructions)

Step 1: Type **edsysadm** and press ENTER.

**NOTE:** If your change requires new files or changes to existing files and you do not execute this command from the directory in which the files reside, enter full pathnames when supplying filenames.

Step 2: You are asked to choose between a menu and a task, Choose **task** and press ENTER.

Step 3: You are asked to choose between adding a new task and changing an existing one. Choose **change** and press ENTER.

Step 4: You are asked if your change is for an on-line or for a task that has been saved for a package. Choose **on-line** and press SAVE.

Step 5: You are asked to supply the name and location of the task to be changed. Fill in the blanks and press SAVE.

- Step 6: You are given a task definition form filled in with the current values for the task named above. Make the desired changes and press SAVE. (See “The Task Definition Form” for descriptions of the fields on the form.)
- Step 7: You are asked if you want to install the modifications into the interface on your machine or save them for a package. Choose **install** and press SAVE.
- Step 8: You see one of the following:
- A confirmation screen if the installation was successful. This means that the change is in place.
  - An error message describing the reason installation was unsuccessful.
  - A message explaining that there is a conflict with the name you have chosen for this menu. You are prompted to choose from three possible actions: install (the addition is installed despite the collision of names), rename or relocate (you are shown the menu form at which time you can change the name or location of the entry), and do not install.

---

### The Task Definition Form

This form contains six fields in which you must provide the following information: a task name, a task description, a task location, the name of a help message for the task, a task action file, and the files associated with the task. Below are descriptions of the information you must provide in each field.

#### Task Name

The name of the new task (as it should appear in the lefthand column of the screen). This field has a maximum length of 16 characters and should consist of alphanumeric characters.

### Task Description

A description of the new task (as it should appear in the right column of the screen). This field has a maximum length of 58 characters and can consist of any alphanumeric character except an at sign (@), circumflex (^), tilde (~), back quote (`), single quote ('), and double quotes (").

### Task Location

The location of the task in the menu followed by all other menus that must be traversed (in the order they are traversed) to access this task. Each menu name must be separated by colons. For example, the task location for a task entry being added to the **Applications** menu is **main:applications**. Do not include the task name in this location definition. The complete path to this task entry is the task location plus the task name defined at the first prompt.

This field has a maximum length of 50 characters and should consist of alphanumeric characters.

### Task Help File Name

Pathname to the item help file for this task entry. If it resides in the directory from which you invoked **edsysadm**, you do not need to give a full pathname. If you name an item help file that does not exist, you are placed in an editor (as defined by **\$EDITOR**) to create one. The new file is created in the current directory and named *Help*.

### Task Action

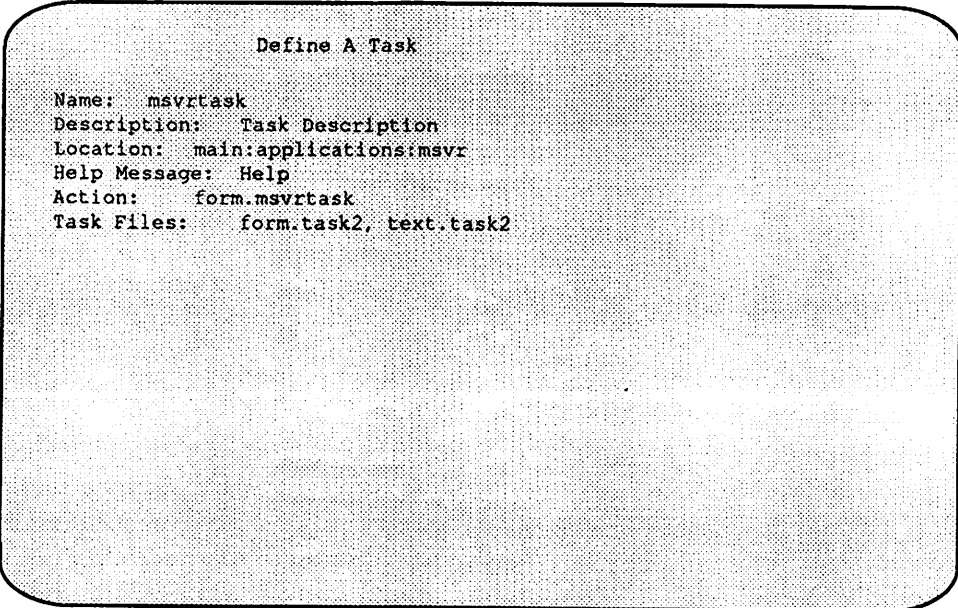
The **FACE** form name or executable that is run when this task is selected. This field has a maximum length of 50 characters and should consist of alphanumeric characters. This pathname can be relative to the current directory as well as absolute. (Refer to the *UNIX SVR4 Programmer's Guide: System Services and Application Packaging Tools* for details on the task action file.)

### Task Files

Any FACE objects or other executables that support the task action listed above and might be called from within that action. Do not include the item help filename or the task action in this list. Pathnames can be relative to the current directory as well as absolute. A dot (.) implies “all files in the current directory” and includes files in subdirectories.

This field is multi-lines and has a maximum length of 50 characters per line. Entries should consist of alphanumeric characters.

Figure C–8 shows a sample task definition form that has been filled in:



The image shows a sample task definition form titled "Define A Task". The form is enclosed in a rounded rectangle with a light gray background. It contains the following text:

```
Define A Task

Name: msvrtask
Description: Task Description
Location: main:applications:msvr
Help Message: Help
Action: form.msvrtask
Task Files: form.task2, text.task2
```

Figure C–8: Sample Task Definition Form

## Deleting a Menu or Task Entry

To delete either a menu or task entry from the interface, execute

```
delsysadm name
```

where *name* is the location of the task or menu in the interface, followed by the menu or task name. For example, to delete a task named **mytask** with the location **main:application:mymenu**, execute

```
delsysadmmain:application:mymenu:mytask
```

Before you can remove an entry, that menu must be empty (contain no submenus or tasks). If it is not, you must use the **-r** option with **delsysadm**. This option requests that, in addition to the named menu, all submenus and tasks located under that menu be removed. For example, to remove **main:application:mymenu** and all submenus and task that reside under it, execute

```
delsysadm -r main:application:mymenu
```

When you use the **-f** option, **delsysadm** checks for dependencies before removing any subentries. (Dependency exists if the menu being removed contains an entry placed there by an application package.) If a dependency is found, you are shown a list of packages that depend on the menu you want to delete and asked whether you want to continue. If you answer yes, the menu and all of its menus and task are removed (even those shown to have dependencies). If you answer no, the menu is not deleted.

### Caution

Use **delsysadm** to remove only those menu or task entries that you have added to the interface with **edsysadm**.

## Using National Characters on the Console

This appendix describes how to configure the NCR System 3000, UNIX SVR4 installation for use with national keyboards and display of national character sets.

This procedure involves the following steps:

|                                              |     |
|----------------------------------------------|-----|
| Changing Line Characteristics to 8 Bit ..... | D-2 |
| Changing Keyboard Mapping .....              | D-4 |
| Changing Screen Mapping .....                | D-6 |



## Changing Line Characteristics to 8 Bit

Step 1: Log in as root.

Step 2: Find the tty label for the local console (VGA console) in */etc/inittab* by finding the line with the device **vt00**. The tty label follows the device. The following example shows a typical local console line in */etc/inittab*:

```
c1:1:respawn:/sbin/vtgetty vt00 9600NP
```

In this example, the tty label is **9600NP**. For the remote console, the tty label can be found in two places, depending on the run level. For run level 1, the tty label can be found in */etc/inittab* by finding the line with the device **term/00**. The following example shows a typical local console line in */etc/inittab*:

```
n1:1:respawn:/sbin/getty term/00 9600HW
```

In this example, the tty label is **9600HW**. For run levels 2 and 3, the tty label can be found by using the **pmadm(1M)** command with the **-l** option to list all the port services defined for the port monitors. Look for the line with the **/dev/term/00** device. The following example shows a typical entry:

```
PMTAG PMTYPE SVCTAG FLGS ID <PMSPECIFIC>
ttymon3 ttymon 00 u root /dev/term/00
 hb - /usr/bin/login - 9600HW - login: - - #
```

In this example, the tty label is **9600HW** for **/dev/term/00**.

**NOTE:** These examples show the entries for the local and remote console as the system is delivered. If these defaults have been modified, please use your new settings.

Step 3: Using an editor, access the */etc/gettydefs* file for vtgetty or getty services (typically */etc/inittab* entries) and the */etc/ttydefs* file for ttymon services (typically those obtained via the **pmadm(1M)** command).

Step 4: Find the entry which defines the console line settings.

The entry looks similar to the following:

```
rconsole<id> # B<baud rate>
```

<id> is either empty or a number, and <baud rate> is the baud rate the console is currently running (usually 9600).

Identify the entry in which the baud rate matches the current baud rate.

Step 5: Remove ISTRIP twice from the settings of that entry.

Step 6: Insert CS8 to the entry twice if it is not already there.

Step 7: Log out.

Step 8: Log in again.

Step 9: Use the following command to verify that **-istrip** and **cs8** are in effect:

```
stty -a
```

## Changing Keyboard Mapping

This section describes how to map the national keyboard to the internal character set chosen. Performing this mapping ensures that the correct characters are entered to the system when you press a key at the keyboard. This is valuable even if the console monitor does not display the character correct as the entered data may be observed from, for example, a terminal.

---

### Before You Start

Verify that **xcp** (Xenix Compatibility Package) is installed on your system by using the following command:

```
pkginfo xcp
```

If **xcp** is not on your system, install the package before you begin to change the keyboard mapping.

---

### Procedure

Step 1: Change the current working directory to somewhere well-defined (a personal home directory or the like).

Step 2: Save a copy of the current keyboard configuration:

```
mapkey -d > keyboard.org
```

Step 3: Copy the original keyboard setting to a new file that can be modified.

```
cp keyboard.org keyboard.new
```

### Step 4: Modify the *keyboard.new* file.

The *keyboard.new* file contains a list of the 127 keyboard keys in order of their scan code. By modifying this file, you can modify the character sent when a certain key is pressed or a certain combination of keys are pressed (such as <SHIFT> <key>).

To modify the file, you may find it helpful to use a scan code chart of the keyboard. Such charts often appear in the keyboard hardware manual. If you are familiar with the characters sent when a certain key is pressed, you may be able to do most of the modifications by locating the character sent in the *keyboard.new* file and substituting it with the character you want to send instead.

**NOTE:** You must enter 8 bit characters in the file from a dumb terminal running vt220 Mode, Multinational over an 8 bit line.

### Step 5: When you have modified the *keyboard.new* file, take it into use by entering the following command at the console:

```
mapkey -V keyboard.new
```

If you have made any mistakes modifying this file, you can always restore the original configuration using the command:

```
mapkey -V keyboard.org
```

or

```
mapkey -V
```

## Changing Screen Mapping

This section describes how to map the internal character set to the character set used in the monitor. By default, the System 3000 monitor is delivered with “code page 437” character PROM. If the internal character set chosen does not match the provided code page, it is likely that not all characters are displayed correctly on the screen. A trade-off is to define the screen mapping either as the best match or with a special character for the characters not available in code page. Another possibility is to change the PROM with one holding the needed characters.

After you change the keyboard mapping, you may need to modify the screen display as well.

Step 1: Save a copy of the original screen display.

```
mapscrn -d > screen.org
```

Step 2: Copy the screen display into a new file to be modified.

```
cp screen.org screen.new
```

Step 3: Modify the *screen.new* file.

The *screen.new* file contains 256 “slots”, one for each character in the internal character set. Each slot contains either a 7 bit ASCII character or an octal number that identifies an 8-bit character in a code page table. All octal numbers in the original *screen.org* are identical to their slot number.

For example, the 128th character slot contains the octal number 200 (=128 decimal), the 129th character slot contains the octal number 201 (=129 decimal), and so on. Consider the original screen file as a “slot to code page character mapping” that maps a character in the internal character set into the code page character set.

To modify the *screen.new* file so that national 8-bit characters are displayed correctly, you must find the slot number that corresponds to that character. Then modify the slot entry to the octal value of the code page character you want to display instead.

Step 4: Make the modifications effective by entering the following command:

```
mapscrn screen.new
```

You can restore the original configuration by using the following command:

```
mapscrn screen.org
```

**NOTE:** Do NOT attempt to call a script which executes the **mapkey** and **mapscrn** commands in a new shell. If the commands are entered in a script, this must be executed in the shell that is executing at the console; that is, the script name must be preceded with a full stop and a space (. ).

# Sendmail – An Internetwork Mail Router

UNIX SVR4 Mailsurr is sthe standard mail transport for the System 3000. Sendmail is available as an alternate mail transport when the **compat** package is installed.

The information in this appendix is taken from the paper *SENDMAIL—An Internetwork Mail Router*, written by Eric Allman of Britton–Lee, Inc., 1919 Addison Street, Suite 105., Berkeley, California 94704.

|                                           |      |
|-------------------------------------------|------|
| Abstract .....                            | E-2  |
| Design Goals .....                        | E-5  |
| Overview .....                            | E-8  |
| Usage and Implementation .....            | E-13 |
| Comparison with other Mail Programs ..... | E-19 |
| Evaluations and Future Plans .....        | E-22 |
| References .....                          | E-25 |

## Abstract

This paper references a number of Requests for Comments (RFCs), which you may want to read for further information. To get copies of RFCs, contact the Network Information Center (NIC). The address of the Network Information Center is:

Network Information Center  
SRI International  
333 Ravenswood Avenue  
Room EJ291  
Menlo Park, CA 94025

Routing mail through a heterogeneous internet presents many new problems. Among the worst of these is that of address mapping. Historically, this has been handled on an *ad hoc* basis. However, this approach has become unmanageable as internets grow.

**Sendmail** acts as a unified “post office” to which all mail can be submitted. Address interpretation is controlled by a production system, which can parse both domain-based addressing and old-style *ad hoc* addresses. The production system is powerful enough to rewrite addresses in the message header to conform to the standards of a number of common target networks, including old (NCP/RFC733) Arpanet, new (TCP/RFC822) Arpanet, UUCP, and Phonetnet. **Sendmail** also implements an SMTP server, message queueing, and aliasing.

**Sendmail** implements a general internetwork mail routing facility, featuring aliasing and forwarding, automatic routing to network gateways, and flexible configuration.



In a simple network, each node has an address, and resources can be identified with a host-resource pair; in particular, the mail system can refer to users using a host-username pair. Host names and numbers have to be administered by a central authority, but usernames can be assigned locally to each host.

In an internet, multiple networks with different characteristics and managements must communicate. In particular, the syntax and semantics of resource identification change. Certain special cases can be handled trivially by *ad hoc* techniques, such as providing network names that appear local to hosts on other networks, as with the Ethernet at Xerox PARC. However, the general case is extremely complex. For example, some networks require point-to-point routing, which simplifies the database update problem since only adjacent hosts must be entered into the system tables, while others use end-to-end addressing. Some networks use a left-associative syntax and others use a right-associative syntax, causing ambiguity in mixed addresses.

Internet standards seek to eliminate these problems. Initially, these standards proposed expanding the address pairs to address triples, consisting of {network, host, resource} triples. Network numbers must be universally agreed upon, and hosts can be assigned locally on each network. The user-level presentation was quickly expanded to address domains, comprised of a local resource identification and a hierarchical domain specification with a common static root. The domain technique separates the issue of physical versus logical addressing. For example, an address of the form *eric@a.cc.berkeley.arpa* describes only the logical organization of the address space.

**Sendmail** is intended to help bridge the gap between the totally *ad hoc* world of networks that know nothing of each other and the clean, tightly-coupled world of unique network numbers. It can accept old arbitrary address syntaxes, resolving ambiguities using heuristics specified by the system administrator, as well as domain-based addressing. It helps guide the conversion of message formats between disparate networks. In short, **sendmail** is designed to assist a graceful transition to consistent internetwork addressing schemes.

Section 1 discusses the design goals for **sendmail**. Section 2 gives an overview of the basic functions of the system. In section 3, details of usage are discussed. Section 4 compares **sendmail** to other internet mail routers, and an evaluation of **sendmail** is given in section 5, including future plans.

## Design Goals

Design goals for **sendmail** include:

- **Compatibility with the existing mail programs**, including Bell version 6 mail, Bell version 7 mail [UNIX83], Berkeley Mail [Shoens79], BerkNet mail [Schmidt79], and hopefully UUCP mail [Nowitz78a, Nowitz78b]. ARPANET mail [Crocker77a, Postel77] was also required.
- **Reliability**, in the sense of guaranteeing that every message is correctly delivered or at least brought to the attention of a human for correct disposal; no message should ever be completely lost.
- **Existing software** to do actual delivery should be used whenever possible. This goal derives as much from political and practical considerations as technical.
- **Easy expansion** to fairly complex environments, including multiple connections to a single network type (such as with multiple UUCP or Ether nets [Metcalf76]). This goal requires consideration of the contents of an address as well as its syntax in order to determine which gateway to use.
- **Configuration should not be compiled into the code**. A single compiled program should be able to run as is at any site (barring such basic changes as the CPU type or the operating system). This seemingly unimportant goal is critical in real life. Besides the simple problems that occur when any program gets recompiled in a different environment, many sites like to “fiddle” with anything that they will be recompiling anyway.

- **Sendmail** must be able to let various groups maintain their own mailing lists, and let individuals specify their own forwarding, without modifying the system alias file.
- Each user should be able to specify which mailer to execute to process mail being delivered for him or her. This feature allows users who are using specialized mailers that use a different format to build their environment without changing the system, and facilitates specialized functions (such as returning an “I am on vacation” message).
- Network traffic should be minimized by batching addresses to a single host where possible, without assistance from the user.

These goals motivated the architecture illustrated in Figure E-1.

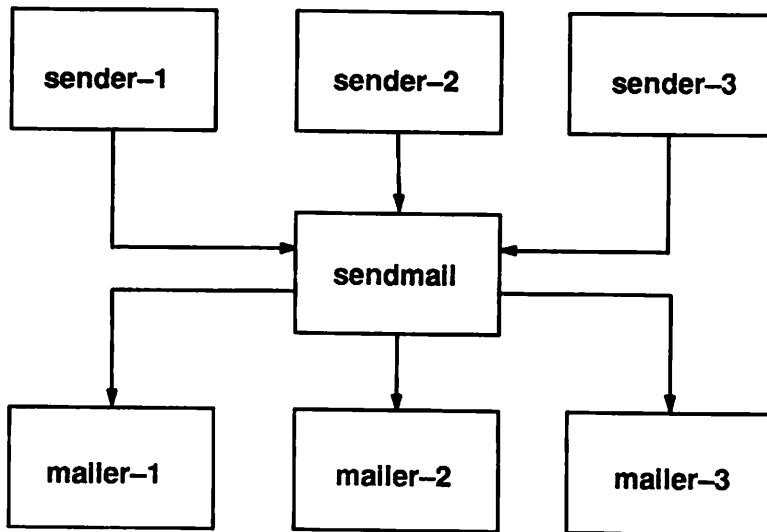


Figure E-1: Sendmail System Structure

The user interacts with a mail generating and sending program. When the mail is created, the generator calls **sendmail**, which routes the message to the correct mailer(s). Since some of the senders may be network servers and some of the mailers may be network clients, **sendmail** may be used as an internet mail gateway.

## Overview

---

### System Organization

**Sendmail** neither interfaces with the user nor does actual mail delivery (unless it is mailing to a file). Rather, it collects a message generated by a user interface program (UIP) such as Berkeley Mail, MS [Crocker77b], or MH [Borden79], edits the message as required by the destination network, and calls appropriate mailers to do mail delivery or queueing for network transmission. This discipline allows the insertion of new mailers at minimum cost. In this sense **sendmail** resembles the Message Processing Module (MPM) of [Postel79b].

---

### Interfaces to the Outside World

**Sendmail** can communicate with the outside world, both in receiving and in sending mail in the following four ways:

- Using the conventional UNIX argument vector/return status
- Speaking SMTP over a pair of UNIX pipes
- Speaking SMTP over an interprocess(or) channel
- Speaking SMTP through the `smtpd` daemon

#### Argument Vector/Exit Status

This technique is the standard UNIX method for communicating with the process. A list of recipients is sent in the argument vector, and the message body is sent on the standard input. Anything that the mailer prints is simply collected and sent back to the sender if there were any problems. The exit status from the mailer is collected after the message is sent, and a diagnostic is printed if appropriate.

## **SMTP Over Pipes**

The SMTP protocol [Postel82] can be used to run an interactive lock-step interface with the mailer. A subprocess is still created, but no recipient addresses are passed to the mailer via the argument list. Instead, they are passed one at a time in commands sent to the processes standard input. Anything appearing on the standard output must be a reply code in a special format.

## **SMTP Over an IPC Connection**

This technique is similar to the previous technique, except that it uses a 4.2bsd IPC channel [UNIX83]. This method is exceptionally flexible in that the mailer need not reside on the same machine. It is normally used to connect to a **sendmail** process on another machine.

## **SMTP Through the smtpd Daemon**

UNIX SVR4 also uses SMTP through the **smtpd** daemon. This technique is the preferred way of using SMTP on a UNIX SVR4 system.

---

## Operational Description

When a sender wants to send a message, it issues a request to **sendmail** using one of the three methods described above. **Sendmail** operates in two distinct phases. In the first phase, it collects and stores the message. In the second phase, message delivery occurs. If errors occur during processing in the second phase, **sendmail** creates and returns a new message describing the error and/or returns a status code telling what went wrong.

### Argument Processing And Address Parsing

If **sendmail** is called using one of the two subprocess techniques, the arguments are first scanned and option specifications are processed. Recipient addresses are then collected, either from the command line or from the SMTP RCPT command, and a list of recipients is created. Aliases are expanded at this step, including mailing lists. As much validation as possible of the addresses is done at this step: syntax is checked, and local addresses are verified, but detailed checking of host names and addresses is deferred until delivery. Forwarding is also performed as the local addresses are verified.

**Sendmail** appends each address to the recipient list after parsing. When a name is aliased or forwarded, the old name is retained in the list, and a flag is set that tells the delivery phase to ignore this recipient. This list is kept free from duplicates, preventing alias loops and duplicate messages delivered to the same recipient, as might occur if a person is in two groups.

### Message Collection

**Sendmail** then collects the message. The message should have a header at the beginning. No formatting requirements are imposed on the message except that they must be lines of text (that is, binary data is not allowed). The header is parsed and stored in memory, and the body of the message is saved in a temporary file.



To simplify the program interface, **sendmail** collects the message even if no addresses were valid and returns the message with an error.

## Message Delivery

For each unique mailer and host in the recipient list, **sendmail** calls the appropriate mailer. Each mailer invocation sends to all users receiving the message on one host. Mailers that only accept one recipient at a time are handled properly. The message is sent to the mailer using one of the same three interfaces used to submit a message to **sendmail**. Each copy of the message is prepended by a customized header. The mailer status code is caught and checked, and a suitable error message given as appropriate. The exit code must conform to a system standard or a generic message ("Service unavailable") is given.

## Queueing For Retransmission

If the mailer returned a status that indicated that it might be able to handle the mail later, **sendmail** queues the mail and tries again later.

## Return To Sender

If errors occur during processing, **sendmail** returns the message to the sender for retransmission. The letter can be mailed back or written in the file *dead.letter* in the sender's home directory.

---

## Message Header Editing

Certain editing of the message header occurs automatically. Header lines can be inserted under control of the configuration file. Some lines can be merged; for example, a "From:" line and a "Full-name:" line can be merged under certain circumstances.

---

## Configuration File

Almost all configuration information is read at runtime from an ASCII file (for example, */usr/ucblib/sendmail.cf*), encoding macro definitions (defining the value of macros used internally), header declarations (telling **sendmail** the format of header lines that it will process specially, that is, lines that it will add or reformat), mailer definitions (giving information such as the location and characteristics of each mailer), and address rewriting rules (a limited production system to rewrite addresses which is used to parse and rewrite the addresses).

To improve performance when reading the configuration file, a memory image can be provided. This provides a “compiled” form of the configuration file. See the **-bz** option for more information.

## Usage and Implementation

---

### Arguments

Arguments may be flags and addresses. Flags set various processing options. Following flag arguments, address arguments may be given, unless we are running in SMTP mode. Addresses follow the syntax in RFC822 [Crocker82] for ARPANET address formats. In brief, the format is:

- Anything in parentheses is thrown away (as a comment).
- Anything in angle brackets (< >) is preferred over anything else. This rule implements the ARPANET standard that addresses of the form *user name <machine-address>* send to the electronic *machine-address* rather than the human *user name*.
- Double quotes ( " ) quote phrases; backslashes ( \ ) quote characters. Backslashes are more powerful in that they cause otherwise equivalent phrases to compare differently — for example, *user* and *"user"* are equivalent, but *\user* is different from either of them.

Parentheses, angle brackets, and double quotes must be properly balanced and nested. The rewriting rules control remaining parsing (except for the special processing done after rewriting local names).

---

### Mail to Files and Programs

Files and programs are legitimate message recipients. Files provide archival storage of messages, useful for project administration and history. Programs are useful as recipients in a variety of situations, for example, to maintain a public repository of systems messages (such as the Berkeley msgs program, or the MARS system [Sattley78]).

Any address passing through the initial parsing algorithm as a local address (that is, not appearing to be a valid address for another mailer) is scanned for two special cases. If prefixed by a vertical bar (`|`), the rest of the address is processed as a shell command. If the user name begins with a slash mark (`/`), the name is used as a file name, instead of a login name.

Files that have `setuid` or `setgid` bits set but no `execute` bits set have those bits honored if **sendmail** is running as root.

---

## Aliasing, Forwarding, Inclusion

**Sendmail** reroutes mail three ways. Aliasing applies system wide. Forwarding allows each user to reroute incoming mail destined for that account. Inclusion directs **sendmail** to read a file for a list of addresses, and is normally used in conjunction with aliasing.

### Aliasing

Aliasing maps names to address lists using a system-wide file `/usr/ucblib/aliases` or as specified in the `sendmail.cf` file. This file is indexed to speed access. Only names that parse as local are allowed as aliases; this guarantees a unique key (since there are no nicknames for the local host).

### Forwarding

After aliasing, recipients that are local and valid are checked for the existence of a *forward* file in their home directory. If it exists, the message is not sent to that user, but rather to the list of users in that file. Often this list contains only one address, and the feature is used for network mail forwarding.

Forwarding also permits a user to specify a private incoming mailer. For example, forwarding to:

```
"|/usr/local/newmail myname"
```

uses a different incoming mailer.

## Inclusion

Inclusion is specified in RFC 733 [Crocker77a] syntax:

```
:Include: pathname
```

An address of this form reads the file specified by pathname and sends to all users listed in that file.

The intent is not to support direct use of this feature, but rather to use this as a subset of aliasing. For example, an alias of the form: `project::include:/usr/project/userlist` is a method of letting a project maintain a mailing list without interacting with the system administration, even if the alias file is protected. It is not necessary to rebuild the index on the alias database when an `:include:` list is changed.

---

## Message Collection

Once all recipient addresses are parsed and verified, the message is collected. The message comes in two parts: a message header and a message body, separated by a blank line.

The header is formatted as a series of lines of the form

```
field-name: field-value
```

Field-value can be split across lines by starting the following lines with a space or a tab. Some header fields have special internal meaning, and have appropriate special processing. Other headers are simply passed through. Some header fields may be added automatically, such as time stamps.

The body is a series of text lines. It is completely uninterpreted and untouched, except that lines beginning with a dot have the dot doubled when transmitted over an SMTP channel. This extra dot is stripped by the receiver.

---

## Message Delivery

The send queue is ordered by the receiving host before transmission in order to implement message batching. Each address is marked as it is sent so rescanning the list is safe. An argument list is built as the scan proceeds. Mail to files is detected during the scan of the send list. The interface to the mailer is performed using one of the techniques described in the “Interfaces to the Outside World” section.

After a connection is established, **sendmail** makes the per-mailer changes to the header and sends the result to the mailer. If any mail is rejected by the mailer, a flag is set to invoke the return-to-sender function after all delivery completes.

---

## Queued Messages

If the mailer returns a “temporary failure” exit status, the message is queued. A control file is used to describe the recipients to be sent to and various other parameters. This control file is formatted as a series of lines, each describing a sender, a recipient, the time of submission, or some other salient parameter of the message. The header of the message is stored in the control file, so that the associated data file in the queue is just the temporary file that was originally collected.

---

## Configuration

Configuration is controlled primarily by a configuration file (usually */usr/ucblib/sendmail.cf*) read at startup. The configuration file encodes macro definitions, header definitions, mailer definitions, rewriting rules, and options. **Sendmail** should not need to be recompiled except

- To change operating systems (V6, V7/32V, 4BSD)
- To remove or insert the DBM (UNIX database) library routines
- To change ARPANET reply codes
- To add headers fields requiring special processing

Adding mailers or changing parsing (that is, rewriting) or routing information does not require recompilation.

If the mail is being sent by a local user, and the user has a local copy of the mailer's configuration in the sender's home directory (for example, **mailx**(1) will look for *\$HOME/.mailrc*), that file is read as a configuration file after the system configuration file. The primary use of this feature is to add header lines. Sendmail is not the primary responsible entity for most header lines, but may add missing lines.

### Macros

Macros can be used in three ways. Certain macros transmit unstructured textual information into the mail system, such as the name and version **sendmail** uses to identify itself in error messages. Other macros transmit information from **sendmail** to the configuration file for use in creating other fields (such as argument vectors to mailers); for example, the name of the sender, and the host and user of the recipient. Other macros are unused internally, and can be used as shorthand in the configuration file.

### Header Declarations

Header declarations inform **sendmail** of the format of known header lines. Knowledge of a few header lines is built into **sendmail**, such as the "From:" and "Date:" lines.

Most configured headers are automatically inserted in the outgoing message if they do not exist in the incoming message. Certain headers are suppressed by some mailers.

### Mailer Declarations

Mailer declarations tell **sendmail** of the various mailers available to it. The definition specifies the internal name of the mailer, the pathname of the program to call, some flags associated with the mailer, and an argument vector to be used on the call; this vector is macro-expanded before use.

## Address Rewriting Rules

The heart of address parsing in **sendmail** is a set of rewriting rules. These are an ordered list of pattern-replacement rules, (somewhat like a production system, except that order is critical), which are applied to each address. The address is rewritten textually until it is either rewritten into a special canonical form (that is, a (mailer, host, user) 3-tuple, such as {arpanet, usc-isif, postel} representing the address "*postel@usc-isif*"), or it falls off the end. When a pattern matches, the rule is reapplied until it fails. The configuration file also supports the editing of addresses into different formats. For example, an address of the form:

```
ucsfcg1!tef
```

might be mapped into:

```
tef@ucsfcg1.UUCP
```

to conform to the domain syntax. Translations can also be done in the other direction.

## Option Setting

Several options can be set from the configuration file. These include the pathnames of various support files, timeouts, default modes, etc.



## Comparison with other Mail Programs

---

### Delivermail

Sendmail is an outgrowth of **delivermail**. The primary differences are:

- Configuration information is not compiled in. This change simplifies many of the problems of moving to other machines. It also allows easy debugging of new mailers.
- Address parsing is more flexible. For example, **delivermail** only supported one gateway to any network, whereas **sendmail** can be sensitive to host names and reroute to different gateways.
- Forwarding and `:include:` features eliminate the requirement that the system alias file be writable by any user (or that an update program be written, or that the system administration make all changes).
- **Sendmail** supports message batching across networks when a message is being sent to multiple recipients.
- A mail queue is provided in **sendmail**. Mail that cannot be delivered immediately but can potentially be delivered later is stored in this queue for a later retry. The queue also provides a buffer against system crashes; after the message has been collected it may be reliably redelivered even if the system crashes during the initial delivery.

- **Sendmail** uses the networking support provided by the Operating System to provide a direct interface networks such as the ARPANET and/or Ethernet using SMTP (the Simple Mail Transfer Protocol) over a TCP/IP connection.

---

## MMDF

**MMDF** [Crocker79] spans a wider problem set than **sendmail**. For example, the domain of **MMDF** includes a “phone network” mailer, whereas **sendmail** calls on preexisting mailers in most cases.

**MMDF** and **sendmail** both support aliasing, customized mailers, message batching, automatic forwarding to gateways, queueing, and retransmission. **MMDF** supports two-stage timeout, which **sendmail** does not support.

The configuration for **MMDF** is compiled into the code. (Dynamic configuration tables are currently being considered for **MMDF**, allowing the installer to select either compiled or dynamic tables.)

Since **MMDF** does not consider backwards compatibility as a design goal, the address parsing is simpler but much less flexible.

It is somewhat harder to integrate a new channel, the **MMDF** equivalent of a **sendmail** mailer, into **MMDF**. In particular, **MMDF** must know the location and format of host tables for all channels, and the channel must speak a special protocol. This allows **MMDF** to do additional verification (such as verifying host names) at submission time.

**MMDF** strictly separates the submission and delivery phases. Although **sendmail** has the concept of each of these stages, they are integrated into one program, whereas in **MMDF** they are split into two programs.

---

## Message Processing Module

The Message Processing Module (MPM) discussed by Postel [Postel79b] matches **sendmail** closely in terms of its basic architecture. However, like **MMDF**, the **MPM** includes the network interface software as part of its domain.

**MPM** also postulates a duplex channel to the receiver, as does **MMDF**, allowing simpler handling of errors by the mailer than is possible in **sendmail**. When a message queued by **sendmail** is sent, any errors must be returned to the sender by the mailer itself. Both **MPM** and **MMDF** mailers can return an immediate error response, and a single error processor can create an appropriate response.

**MPM** prefers passing the message as a structured object, with type-length-value tuples. (This is similar to the NBS standard.) Such a convention requires a much higher degree of cooperation between mailers than is required by **sendmail**. **MPM** also assumes a universally agreed upon internet name space (with each address in the form of a net-host-user tuple), which **sendmail** does not.

## Evaluations and Future Plans

**Sendmail** is designed to work in a nonhomogeneous environment. Every attempt is made to avoid imposing unnecessary constraints on the underlying mailers. This goal has driven much of the design. One of the major problems has been the lack of a uniform address space, as postulated in [Postel79a] and [Postel79b].

A nonuniform address space implies that a path will be specified in all addresses, either explicitly (as part of the address) or implicitly (as with implied forwarding to gateways). This restriction has the unpleasant effect of making replying to messages exceedingly difficult, since there is no one “address” for any person, but only a way to get there from wherever you are.

Interfacing to mail programs that were not initially intended to be applied in an internet environment has been amazingly successful, and has reduced the job to a manageable task.

**Sendmail** has knowledge of a few difficult environments built in. It generates ARPANET FTP/SMTP compatible error messages (prefixed with three-digit numbers [Neigus73, Postel74, Postel82]) as necessary, optionally generates UNIX-style “From” lines on the front of messages for some mailers, and knows how to parse the same lines on input. Also, error handling has an option customized for BerkNet.

The decision to avoid doing any type of delivery where possible (even, or perhaps especially, local delivery) has turned out to be a good idea. Even with local delivery, issues such as the location of the mailbox, the format of the mailbox, the locking protocol used, etc., are best decided by other programs. One surprisingly major annoyance in many internet mailers is that the location and format of local mail is built in. The feeling seems to be that local mail is so common that it should be efficient. This feeling is not born out by our experience; on the contrary, the location and format of mailboxes seems to vary widely from system to system.

The ability to automatically generate a response to incoming mail (by forwarding mail to a program) seems useful (“I am on vacation until late August....”) but can create problems such as forwarding loops (two people on vacation whose programs send notes back and forth, for instance) if these programs are not well written. A program could be written to do standard tasks correctly, but this would solve the general case.

It might be desirable to implement some form of load limiting. I am unaware of any mail system that addresses this problem, nor am I aware of any reasonable solution at this time.

The configuration file is currently practically inscrutable; considerable convenience could be realized with a higher-level format.

It seems clear that common protocols will be changing soon to accommodate changing requirements and environments. These changes will include modifications to the message header (for example, [NBS80]) or to the body of the message itself (such as for multimedia messages [Postel80]). Experience indicates that these changes should be relatively trivial to integrate into the existing system.

In tightly coupled environments, it would be nice to have a name server such as Grapevine [Birrell82] integrated into the mail system. This would allow a site such as “Berkeley” to appear as a single host, rather than as a collection of hosts, and would allow people to move transparently among machines without having to change their addresses. Such a facility would require an automatically updated database and some method of resolving conflicts. Ideally this would be effective even without all hosts being under a single management. However, it is not clear whether this feature should be integrated into the aliasing facility or should be considered a “value added” feature outside **sendmail** itself. As a more interesting case, the CSNET name server [Solomon81] provides a facility that goes beyond a single tightly-coupled environment. Such a facility would normally exist outside of **sendmail** however.

## References

[Birrell82]

Birrell, A. D., Levin, R., Needham, R. M., and Schroeder, M. D., "Grapevine: An Exercise in Distributed Computing." In *Communications of the ACM* 25, 4, April 82.

[Borden79]

Borden, S., Gaines, R. S., and Shapiro, N. Z., *The MH Message Handling System: Users' Manual*. R-2367-PAF. Rand Corporation. October 1979.

[Crocker77a]

Crocker, D. H., Vittal, J. J., Pogran, K. T., and Henderson, D. A. Jr., *Standard for the Format of ARPA Network Text Messages*. RFC 733, NIC 41952. In [Feinler78]. November 1977.

[Crocker77b]

Crocker, D. H., *Framework and Functions of the MS Personal Message System*. R-2134-ARPA, Rand Corporation, Santa Monica, California. 1977.

[Crocker79]

Crocker, D. H., Szurkowski, E. S., and Farber, D. J., *An Internetwork Memo Distribution Facility—MMDF*. 6th Data Communication Symposium, Asilomar. November 1979.

[Crocker82]

Crocker, D. H., *Standard for the Format of ARPA Internet Text Messages*. RFC 822. Network Information Center, SRI International, Menlo Park, California. August 1982.

[Metcalf76]

Metcalf, R., and Boggs, D., “Ethernet: Distributed Packet Switching for Local Computer Networks”, *Communications of the ACM* 19, 7. July 1976.

[Feinler78]

Feinler, E., and Postel, J. (eds.), *ARPANET Protocol Handbook*. NIC 7104, Network Information Center, SRI International, Menlo Park, California. 1978.

[NBS80]

National Bureau of Standards, *Specification of a Draft Message Format Standard*. Report No. ICST/CBOS 80-2. October 1980.

[Neigus73]

Neigus, N., *File Transfer Protocol for the ARPA Network*. RFC 542, NIC 17759. In

[Feinler78].

August, 1973.

[Nowitz78a]

Nowitz, D. A., and Lesk, M. E., *A Dial-Up Network of UNIX Systems*. Bell Laboratories. In *UNIX Programmer's Manual*, Seventh Edition, Volume 2. August, 1978.

[Nowitz78b]

Nowitz, D. A., *UUCP Implementation Description*. Bell Laboratories. In *UNIX Programmer's Manual*, Seventh Edition, Volume 2. October, 1978.

[Postel74]

Postel, J., and Neigus, N., *Revised FTP Reply Codes*. RFC 640, NIC 30843. In [Feinler78]. June, 1974.

[Postel77]

Postel, J., *Mail Protocol*. NIC 29588. In



[Feinler78].

November 1977.

[Postel79a]

Postel, J., *Internet Message Protocol*. RFC 753, IEN 85.

Network Information Center, SRI International, Menlo Park, California. March 1979.

[Postel79b]

Postel, J. B., *An Internetwork Message Structure*. In *Proceedings of the Sixth Data Communications Symposium*, IEEE. New York. November 1979.

[Postel80]

Postel, J. B., *A Structured Format for Transmission of Multi-Media Documents*. RFC 767. Network Information Center, SRI International, Menlo Park, California. August 1980.

[Postel82]

Postel, J. B., *Simple Mail Transfer Protocol*. RFC821 (obsoleting RFC788). Network Information Center, SRI International, Menlo Park, California. August 1982.

[Schmidt79]

Schmidt, E., *An Introduction to the Berkeley Network*. University of California, Berkeley California. 1979.

[Shoens79]

Shoens, K., *Mail Reference Manual*. University of California, Berkeley. In *UNIX Programmer's Manual*, Seventh Edition, Volume 2C. December 1979.

[Sluizer81]

Sluizer, S., and Postel, J. B., *Mail Transfer Protocol*. RFC 780. Network Information Center, SRI International, Menlo Park, California. May 1981.

**[Solomon81]**

Solomon, M., Landweber, L., and Neuhengen, D., “The Design of the CSNET Name Server.” CS-DN-2, University of Wisconsin, Madison. November 1981.

**[Su82]**

Su, Zaw-Sing, and Postel, Jon, *The Domain Naming Convention for Internet User Applications*. RFC819. Network Information Center, SRI International, Menlo Park, California. August 1982.

**[UNIX83]**

*The UNIX Programmer's Manual*, Seventh Edition, Virtual VAX-11 Version, Volume 1. Bell Laboratories, modified by the University of California, Berkeley, California. March, 1983.

# **Sendmail Installation and Operation**

The error/status messages in this appendix are taken from RFC 821. The other information is taken from the paper *SENDMAIL Installation and Operation Guide*, written by Eric Allman of Britton-Lee, Inc., 1919 Addison Street, Suite 105, Berkeley, California 94704.

|                                |      |
|--------------------------------|------|
| Introduction .....             | F-2  |
| Basic Installation .....       | F-4  |
| Normal Operations .....        | F-10 |
| Arguments .....                | F-19 |
| Tuning .....                   | F-21 |
| The Configuration File .....   | F-27 |
| Command Line Flags .....       | F-48 |
| Configuration Options .....    | F-50 |
| Mailer Flags .....             | F-55 |
| Other Configuration .....      | F-58 |
| Summary of Support Files ..... | F-66 |
| Error/Status Messages .....    | F-68 |

## Introduction

This appendix references a number of Requests for Comments (RFCs), which you may want to read for further information. To get copies of RFCs, contact the Network Information Center (NIC). The address of the Network Information Center is:

Network Information Center  
SRI International  
333 Ravenswood Avenue  
Room EJ291  
Menlo Park, CA 94025

**Sendmail** implements a general purpose internetwork mail routing facility under the UNIX operating system. It is not tied to any one transport protocol — its function may be likened to a crossbar switch, relaying messages from one domain into another. In the process, it can do a limited amount of message header editing to put the message into a format that is appropriate for the receiving domain. All of this is done under the control of a configuration file.

**NOTE:** The **sendmail** configuration file */usr/ucblib/sendmail.cf* is not configured for you. You must make any changes.

Due to the requirements of flexibility for **sendmail**, the configuration file can seem somewhat unapproachable. However, there are only a few basic configurations for most sites, for which standard configuration files have been supplied. Most other configurations can be built by adjusting an existing configuration files incrementally.

Although **sendmail** is intended to run without being monitored, it has a number of features that may be used to monitor or adjust the operation under unusual circumstances. These features are described.

Section one describes how to do a basic **sendmail** installation. Section two explains the day-to-day information you should know to maintain your mail system. If you have a relatively normal site, these two sections should contain sufficient information for you to install **sendmail** and keep it happy. Section three describes some parameters that may be safely tweaked. Section four contains information about the command line arguments. Section five contains detailed information about the configuration file. This section is for people who must write their own configuration file. The other sections give a brief but detailed explanation of a number of features not described in the rest of the appendix.

The references in this appendix are actually found in the companion paper *SENDMAIL — An Internetwork Mail Router* (Appendix E). You should read Appendix E before reading this manual to gain a basic understanding of how the pieces fit together.

## Basic Installation

There are two basic steps to installing **sendmail**. The hard part is building the configuration table. This file, which **sendmail** reads when it starts up, describes the mailers it knows about, how to parse addresses, how to rewrite the message header, and the settings of various options. Although the configuration table is quite complex, you can usually build a configuration by adjusting an existing off-the-shelf configuration. The second step is actually doing the installation, that is, creating the necessary files, etc.

**NOTE:** The NCR System 3000 Operating System already has a copy of the **sendmail** program and *sendmail.cf* file installed. There are essentially no other support or source files supplied. You may choose to obtain a later version of the **sendmail** source from a public source such as USENET.

The remainder of this section describes the installation of **sendmail**. It assumes you can use one of the existing configurations and that the standard installation parameters are acceptable. All pathnames and examples are given from the root of the **sendmail** sub-tree, normally */usr/src/usr.lib/sendmail* on 4.3BSD. The NCR System 3000 uses */usr/ucblib/sendmail*. Some pathnames can be configured within the */usr/ucblib/sendmail.cf* file; this file is a system's reference authority.

## Off-The-Shelf Configurations

Configuration files currently in use at Berkeley are in the directory *cf* of the *sendmail* directory. This directory contains three subdirectories: *cf*, *m4*, and *sitedep*. The directory *cf/m4* contains site independent *m4(1)* include files that have information common to all configuration files, while *cf/sitedep* contains *m4(1)* include files that have site-specific information in them. These files are used by the master configuration (*.mc*) in *cf/cf* and produce standard configuration files (with *.cf* suffix) when run through *m4(1)*.

Three off the shelf configurations are supplied to handle the basic cases:

- Internet sites running the nameserver (or using host tables wherein the fully-qualified domain name of each host is listed first) can use *cf/tcppproto.cf*. For simple sites, you should be able to use this file without modifying it. This file is not in *m4* format.
- UUCP only sites can use *cf/uucppproto.cf*. This file is not in *m4* format.
- A group of machines at a single site connected by an ethernet (or other networking that supports TCP/IP) with (only) one host connected to the outside world via UUCP is represented by two configuration files: *cf/tcpuucppproto.cf* should be installed on the host with outside connections and *cf/tcppproto.cf* should be installed on all other hosts.

Some configuration is needed in each of the above cases. Just be sure to correctly fill in the “blanks” as shown in the instructions in the configuration file. Then install the file as */usr/ucblib/sendmail.cf*.

If you are running a larger or more complex site, it is to your advantage to read the *README* file in the *cf* subdirectory. This file explains how to use *m4(1)* to automatically create configuration files for non-standard situations.

---

### Installation Using the Makefile

A makefile exists in the root of the *sendmail* directory that does all of these steps for a 4.3BSD system. You may have to tailor it slightly to use it on other systems.

Before using this makefile, you should perform the following steps:

- Create a symbolic link from *cf* to the directory containing your configuration files.
- Create your configuration file and leave it in the file *cf/system.cf* where *system* is the name of your system (that is, what is returned by *hostname(1)*). If you do not have *hostname*, you can use the declaration “HOST=system” on the *make(1)* command line.
- Examine the file *md/config.m4* and change the m4 macros there to reflect any libraries and compilation flags you may need.

The basic installation procedure is to type the following commands in the root directory of the *sendmail* distribution:

```
make
make install
make installcf
```

These commands make all binaries and install them in the standard places. You must have root permissions to issue the second and third make commands.

---

### Installation by Hand

Along with building a configuration file, you must install the *sendmail* startup into your UNIX system. If you are doing this installation in conjunction with a regular Berkeley UNIX install, these steps will already be complete. You must have root permissions to perform many of these steps.



### **/usr/ucblib/sendmail**

The binary for **sendmail** is located in */usr/ucblib*. If you need to recompile and reinstall the entire system, enter the following sequence:

```
cd src
make clean
make install
```

### **/usr/ucblib/sendmail.cf**

Install the configuration file that you created earlier in */usr/ucblib/sendmail.cf*:

```
cp cf/system.cf /usr/ucblib/sendmail.cf
```

### **/usr/ucb/newaliases**

If you are running **delivermail**, it is critical that you replace the **newaliases** command. This can just be a link to **sendmail**:

```
rm -f /usr/ucb/newaliases
ln /usr/ucblib/sendmail /usr/lib/newaliases
```

### **/usr/spool/mqueue**

Create the directory */usr/ucblib/mqueue* to hold the mail queue. This directory should be mode 755 and owned by root.

### **/usr/ucblib/aliases\***

The system aliases are held in three files. The file */usr/ucblib/aliases* is the master copy. A sample is given in *lib/aliases* which includes some aliases which must be defined:

```
cp lib/aliases /usr/ucblib/aliases
```

You should extend this file with any aliases that are apropos to your system.

Normally **sendmail** looks at a version of these files maintained by the **dbm(3)** routines. These are stored in */usr/ucblib/aliases.dir* and */usr/ucblib/aliases.pag*. You can initially create these as empty files, but you must initialize them promptly. These should be mode **644**:

```
cp /dev/null /usr/ucblib/aliases.dir
cp /dev/null /usr/ucblib/aliases.pag
chmod 644 /usr/ucblib/aliases.*
newaliases
```

### **/usr/ucblib/sendmail.fc**

If you intend to install the frozen version of the configuration file (for quick startup), you should create the file */usr/ucblib/sendmail.fc* and initialize it. This step may be safely skipped.

```
cp /dev/null /usr/ucblib/sendmail.fc
/usr/ucblib/sendmail -bz
```

### **/etc/init.d/win**

It will be necessary to start up the **sendmail** daemon when your system reboots. This daemon performs two functions: it listens on the SMTP socket for connections (to receive mail from a remote system) and it processes the queue periodically to ensure that mail gets delivered when hosts come up.

The following lines should be added to a script in */etc/init.d/win* to start up the daemons:

```
if [-f /usr/ucblib/sendmail]; then
(cd /usr/ucblib/mqueue; rm -f [lnx]f*)
/usr/ucblib/sendmail -bd -q30m &
fi
```

The **cd** and **rm** commands ensure that all lock files have been removed; extraneous lock files may be left around if the system goes down in the middle of processing a message. The line that actually invokes **sendmail** has two flags: **-bd** causes it to listen on the SMTP port, and **-q30m** causes it to run the queue every half hour.

If you are not running a version of UNIX that supports Berkeley TCP/IP, do not include the **-bd** flag. The NCR System 3000 supports Berkeley sockets, and therefore uses this flag.

### **/usr/ucblib/sendmail.hf**

This is the help file used by the SMTP HELP command. Copy it from *lib/sendmail.hf*:

```
cp lib/sendmail.hf /usr/ucblib
```

### **/usr/ucblib/sendmail.st**

If you wish to collect statistics about your mail traffic, you should create the file */usr/ucblib/sendmail.st*:

```
cp /dev/null /usr/ucblib/sendmail.st
chmod 666 /usr/ucblib/sendmail.st
```

This file does not grow. It is printed with the program */usr/ucb/mailstats*.

### **/usr/ucb/newaliases**

If **sendmail** is invoked as “newaliases,” it simulates the **-bi** flag (that is, it rebuilds the alias database). This program must be a link to */usr/ucblib/sendmail*.

### **/usr/ucb/mailq**

If **sendmail** is invoked as “mailq,” it simulates the **-bp** flag (that is, **sendmail** prints the contents of the mail queue). This program must be a link to */usr/ucblib/sendmail*.

## Normal Operations

---

### Quick Configuration Startup

To set up a fast version of the configuration file, use the **-bz** flag:

```
/usr/ucblib/sendmail -bz
```

This command creates the file */usr/ucblib/sendmail.fc* (frozen configuration). This file is an image of sendmail data space after reading in the configuration file. If this file exists, it is used instead of */usr/ucblib/sendmail.cf*. *sendmail.fc* must be rebuilt manually every time *sendmail.cf* is changed.

The frozen configuration file is ignored if a **-C** flag is specified or if **sendmail** detects that it is out of date. However, the heuristics are not strong, so this should not be trusted.

---

### The System Log

The system log is supported by the **syslogd(1M)** program.

#### Format

Each line in the system log consists of a timestamp, the name of the machine that generated it (for logging from several machines over the local area network), the word “**sendmail:**”, and a message.

## Levels

If you have **syslogd**(1M) or an equivalent installed, you are able to do logging. A large amount of information can be logged. The log is arranged as a succession of levels. At the lowest level, only extremely strange situations are logged. At the highest level, even the most mundane and uninteresting events are recorded for posterity. As a convention, log levels under ten are considered “useful;” log levels above ten are usually for debugging purposes.

A complete description of the log levels appears in the “Tuning” section. A **syslogd**(1M) man page is also available.

---

## The Mail Queue

The mail queue should be processed transparently. However, you may find that manual intervention is sometimes necessary. For example, if a major host is down for a period of time the queue may become clogged. Although **sendmail** ought to recover gracefully when the host comes up, you may find performance unacceptably bad in the meantime.

## Printing the Queue

You can print the contents of the queue using the **mailq** command (or by specifying the **-bp** flag to **sendmail**):

```
mailq
```

This command produces a list of queue identification numbers, the size of the message, the date the message entered the queue, and the sender and recipients.

## Format of Queue Files

All queue files have the form *xfAA99999* where *AA99999* is the ID for this file and the *x* is a type. The types are:

**d**

The data file. The message body (excluding the header) is kept in this file.

**l**

The lock file. If this file exists, the job is currently being processed, and a queue run will not process the file. For that reason, an extraneous *lf* file can cause a job to apparently disappear (it will not even time out!).

**n**

This file is created when an ID is being created. It is a separate file to ensure that no mail can ever be destroyed due to a race condition. It should exist for no more than a few milliseconds at any given time.

**q**

The queue control file. This file contains the information necessary to process the job.

**t**

A temporary file. This is an image of the *qf* file when it is being rebuilt. It should be renamed to a *qf* file very quickly.

**x**

A transcript file, existing during the life of a session showing everything that happens during that session.

The *qf* file is structured as a series of lines each beginning with a code letter. The lines are as follows:

**D**

The name of the data file. There may only be one of these lines.

**H**

A header definition. There may be any number of these lines. The order is important: they represent the order in the final message. These use the same syntax as header definitions in the configuration file.

**R**

A recipient address. The recipient address is normally a complete alias, but is actually converted to another alias when the job is processed. There is one line for each recipient.

**S**

The sender address. There may only be one of these lines.

**E**

An error address. If any such lines exist, they represent the addresses that should receive error messages.

**T**

The job creation time. This is used to compute when to time out the job.

**P**

The current message priority. This is used to order the queue. Higher numbers mean lower priorities. The priority changes as the message sits in the queue. The initial priority depends on the message class and the size of the message.

**M**

A message. This line is printed by the mailq command, and is generally used to store status information. It can contain any text.

As an example, the following is a queue file sent to *mckusick@calder* and *wnj*:

```
DdfA13557
Seric
T404261372
P132
Rmckusick@calder
Rwnj
H?D?date: 23-Oct-82 15:49:32-PDT (Sat)
H?F?from: eric (Eric Allman)
H?x?full-name: Eric Allman
Hsubject: this is an example message
Hmessage-id:<8209232249.13557@UCBARPA.BERKELEY.EDU>
Hreceived:by UCBARPA.BERKELEY.EDU (3.227 [10/22/82])
 id A13557; 23-Oct-82 15:49:32-PDT (Sat)
HTo: mckusick@calder, wnj
```

This example shows the name of the data file, the person who sent the message, the submission time (in seconds since January 1, 1970), the message priority, the message class, the recipients, and the headers for the message.

### Forcing the queue

**Sendmail** should run the queue automatically at intervals (refer to `-q` and to “Queue Interval” under the “Arguments” section). The algorithm is to read and sort the queue, and then to attempt to process all jobs in order. When it attempts to run the job, **sendmail** first checks to see if the job is locked. If so, it ignores the job.

There is no attempt to ensure that only one queue processor exists at any time, since there is no guarantee that a job cannot take forever to process. Due to the locking algorithm, it is impossible for one job to freeze the queue. However, an uncooperative recipient host or a program recipient that never returns can accumulate many processes in your system. Unfortunately, there is no way to resolve this without violating the protocol.



In some cases, you may find that a major host going down for a couple of days may create a prohibitively large queue. This results in **sendmail** spending an inordinate amount of time sorting the queue. This situation can be fixed by moving the queue to a temporary place and creating a new queue. The old queue can be run later when the offending host returns to service.

To do this, it is acceptable to move the entire queue directory (as root):

```
cd /usr/ucblib
mv mqueue omqueue; mkdir mqueue; chmod 755 mqueue
```

You should then kill the existing daemon (since it is still processing in the old queue directory) and create a new daemon.

To run the old mail queue, run the following command:

```
/usr/ucblib/sendmail -oQ/usr/ucblib/omqueue -q
```

The **-oQ** flag specifies an alternate queue directory and the **-q** flag says to just run every job in the queue. You can use the **-v** flag to watch what is going on.

When the queue is finally emptied, you can remove the directory:

```
rmdir /usr/ucblib/omqueue
```

---

## The Alias Database

The alias database exists in two forms. One is a text form, maintained in the file `/usr/ucblib/aliases`. The aliases are of the form

```
name: name1, name2, ...
```

Only local names are converted to aliases; for example,

```
eric@mit-xx: eric@berkeley.EDU
```

does not have the desired effect. Aliases may be continued by starting any continuation lines with a space or a tab. Blank lines and lines beginning with a sharp sign (#) are comments.

The second form is processed by the **dbm(3)** library. This form is in the files */usr/ucblib/aliases.dir* and */usr/ucblib/aliases.pag*. This is the form that **sendmail** actually uses to resolve aliases. This technique is used to improve performance.

### Rebuilding the Alias Database

**NOTE:** The System 3000 does not support the DBM library.

You can rebuild the DBM version of the database explicitly by executing the command

```
newaliases
```

This is equivalent to giving **sendmail** the **-bi** flag:

```
/usr/ucblib/sendmail -bi
```

If the **D** option is specified in the configuration, **sendmail** rebuilds the alias database automatically if possible when it is out of date. The conditions under which it does this are:

- The DBM version of the database is mode 666.
- **Sendmail** is running setuid to root.

Auto-rebuild can be dangerous on heavily loaded machines with large alias files; if it might take more than five minutes to rebuild the database, there is a chance that several processes will start the rebuild process simultaneously.

### Potential problems

A number of problems can occur with the alias database. They all result from a **sendmail** process accessing the DBM version while it is only partially built. This can happen under two circumstances: One process accesses the database while another process is rebuilding it, or the process rebuilding the database dies (due to being killed or a system crash) before completing the rebuild.

**Sendmail** has two techniques to try to relieve these problems. First, it ignores interrupts while rebuilding the database; this avoids the problem of someone aborting the process leaving a partially rebuilt database. Second, at the end of the rebuild, it adds an alias of the form

@: @

(which is not normally legal). Before **sendmail** accesses the database, it checks to ensure that this entry exists. (The **a** option is required in the configuration for this action to occur. This should normally be specified unless you are running **delivermail** in parallel with **sendmail**.) **Sendmail** waits for this entry to appear, at which point it forces a rebuild itself.

**NOTE:** The **D** option must be specified in the configuration file for this operation to occur. If the **D** option is not specified, a warning message is generated and **sendmail** continues.

## List Owners

If an error occurs on sending to a certain address, say *x*, **sendmail** looks for an alias of the form “owner-*x*” to receive the errors. This is typically useful for a mailing list where the originator of the list has no control over the maintenance of the list itself; in this case, the list maintainer would be the owner of the list. For example:

```
unix-wizards: eric@ucbarpa, wnj@monet,
nosuchuser, sam@mat.isse
owner-unix-wizards: eric@ucbarpa
```

would cause *eric@ucbarpa* to get the error that occurs when someone sends to *unix-wizards* due to the inclusion of *nosuchuser* on the list.

---

### Per-User Forwarding (.forward Files)

As an alternative to the alias database, any user may put a file with the name *forward* in his or her home directory. If this file exists, **sendmail** redirects mail for that user to the list of addresses listed in the *forward* file. For example, if the home directory for user *mckusick* has a *forward* file with contents:

```
mckusick@ernie
kirk@calder
```

then any mail arriving for *mckusick* is redirected to the specified accounts.

---

### Special Header Lines

Several header lines have special interpretations defined by the configuration file. Others have interpretations built into **sendmail** that cannot be changed without changing the code. These built-ins are described here.

#### Return-Receipt-To:

If this header is sent, a message is sent to any specified addresses when the final delivery is complete, that is, when successfully delivered to a mailer with the *l* flag (local delivery) set in the mailer descriptor.

#### Errors-To:

If errors occur anywhere during processing, this header causes error messages to go to the listed addresses rather than to the sender. This is intended for mailing lists.

#### Apparently-To:

If a message comes in with no recipients listed in the message (in a *To:*, *Cc:*, or *Bcc:* line) then **sendmail** adds an *Apparently-To:* header line for any recipients it is aware of. This is not put in as a standard recipient line to warn any recipients that the list is not complete.

At least one recipient line is required under RFC 822.

## Arguments

The complete list of arguments to **sendmail** appears in the “Command Line Flags” section. Some important arguments are described here.

---

### Queue Interval

The amount of time between forking a process to run through the queue is defined by the **-q** flag. If you run in mode **f** or **a**, this can be relatively large, since it is only relevant when a host that was down comes back up. If you run in **q** mode, it should be relatively short, since it defines the maximum amount of time that a message may sit in the queue.

---

### Daemon Mode

If you allow incoming mail over an IPC connection, you should have a daemon running. This should be set by your */etc/init.d* startup script file using the **-bd** flag. The **-bd** flag and the **-q** flag may be combined in one call:

```
/usr/ucblib/sendmail -bd -q30m &
```

---

### Forcing the Queue

In some cases, you may find that the queue has gotten clogged for some reason. You can force a queue run using the **-q** flag (with no value). It is entertaining to use the **-v** flag (verbose) when this is done to watch what happens:

```
/usr/ucblib/sendmail -q -v
```

---

### Debugging

A fairly large number of debug flags are built into **sendmail**. Each debug flag has a number and a level, where higher levels means to print out more information. The convention is that levels greater than nine are “absurd,” that is, they print out so much information that you would not normally want to see them except for debugging that particular piece of code.

Debug flags are set using the **-d** option; the syntax is:

```
debug-flag: -d debug-list
debug-list: debug-option [, debug-option]
debug-option: debug-range [. debug-level]
debug-range: integer | integer integer
debug-level: integer
```

where spaces are for reading ease only. For example,

```
-d12 Set flag 12 to level 1
-d12.3 Set flag 12 to level 3
-d3-17 Set flags 3 through 17 to level 1
-d3-17.4 Set flags 3 through 17 to level 4
```

For a complete list of the available debug flags, look at a publicly-distributed version of the code (they are too dynamic to keep this documentation up to date).

---

### Trying a Different Configuration File

You can specify an alternative configuration file using the **-C** flag; for example,

```
/usr/ucblib/sendmail -Ctest.cf
```

uses the configuration file *test.cf* instead of the default */usr/lib/sendmail.cf*. If the **-C** flag has no value it defaults to *sendmail.cf* in the current directory.

---

### Changing the Values of Options

To override options, use the **-o** flag. For example,

```
/usr/ucblib/sendmail -oT2m
```

sets the T (timeout) option to two minutes for this run only.

## Tuning

You may want to change a number of configuration parameters , depending on the requirements of your site. Most of these are set using an option in the configuration file. For example, the line "OT3d" sets option T to the value "3d" (three days).

Most of these options default appropriately for most sites. However, sites having very high mail loads may find they need to tune them as appropriate for their mail load. In particular, sites experiencing a large number of small messages, many of which are delivered to many recipients, may find that they need to adjust the parameters dealing with queue priorities.

---

### Timeouts

All time intervals are set using a scaled syntax. For example, "10m" represents ten minutes, whereas "2h30m" represents two and a half hours. The full set of scales is:

- s — seconds
- m — minutes
- h — hours
- d — days
- w — weeks

### Queue interval

The argument to the -q flag specifies how often a sub-daemon runs the queue. This is typically set to between fifteen minutes and one hour.

## Read timeouts

It is possible to time out when reading the standard input or when reading from a remote SMTP server. Technically, this is not acceptable within the published protocols. However, it might be appropriate to set it to something large in certain environments (such as an hour). This reduces the chance of large numbers of idle daemons piling up on your system. This timeout is set using the **r** option in the configuration file.

## Message timeouts

After sitting in the queue for a few days, a message times out. This ensures that at least the sender is aware of the inability to send a message. The timeout is typically set to three days. This timeout is set using the **T** option in the configuration file.

The time of submission is set in the queue, rather than the amount of time left until timeout. As a result, you can flush messages that have been hanging for a short period by running the queue with a short message timeout. For example,

```
/usr/ucblib/sendmail -oTld -q
```

runs the queue and flushes anything that is one day old.

---

## Forking During Queue Runs

If you set the **Y** option, **sendmail** forks before each individual message while running the queue. This prevents **sendmail** from consuming large amounts of memory, so it may be useful in memory-poor environments. However, if the **Y** option is not set, **sendmail** keeps track of hosts that are down during a queue run, which can improve performance dramatically.



## Queue Priorities

Every message is assigned a priority when it is first instantiated, consisting of the message size (in bytes) offset by the message class times the “work class factor” and the number of recipients times the “work recipient factor.” The priority plus the creation time of the message (in seconds since January 1, 1970) are used to order the queue. Higher numbers for the priority mean that the message will be processed later when running the queue.

The message size is included so that large messages are penalized relative to small messages. The message class allows users to send “high priority” messages by including a “Precedence:” field in their message; the value of this field is looked up in the **P** lines of the configuration file. Since the number of recipients affects the amount of load a message presents to the system, this is also included into the priority.

You can set the recipient and class factors in the configuration file using the **y** and **z** options respectively. They default to 1000 (for the recipient factor) and 1800 (for the class factor). The initial priority is:

$$\text{pri} = \text{size} \quad (\text{class} * \text{z}) + (\text{nrcpt} * \text{y})$$

(Remember, higher values for this parameter actually mean that the job has lower priority.)

The priority of a job can also be adjusted each time it is processed (that is, each time an attempt is made to deliver it) using the “work time factor,” set by the **Z** option. This is added to the priority, so it normally decreases the precedence of the job, on the grounds that jobs that have failed many times will tend to fail again in the future.

---

## Load Limiting

**Sendmail** can be asked to queue (but not deliver) mail if the system load average gets too high, using the **x** option. When the load average exceeds the value of the **x** option, the delivery mode is set to **q** (queue only) if the Queue Factor (**q** option) divided by the difference in the current load average and the **x** option plus one exceeds the priority of the message. That is, the message is queued if:

$$pri > QF / ( LA - x + 1 )$$

The **q** option defaults to 10000, so each point of load average is worth 10000 priority points (as described above, that is, bytes + seconds + offsets).

For drastic cases, the **X** option defines a load average at which **sendmail** will refuse to accept network connections. Locally generated mail (including incoming UUCP mail) is still accepted.

---

## Delivery Mode

**Sendmail** can operate in a number of delivery modes, set by the **d** configuration option. These modes specify how quickly mail is delivered. Legal modes are:

- **i** — deliver interactively (synchronously)
- **b** — deliver in background (asynchronously)
- **q** — queue only (don't deliver)

There are trade offs. Mode **i** passes the maximum amount of information to the sender, but is hardly ever necessary. Mode **q** puts the minimum load on your machine, but means that delivery may be delayed for up to the queue interval. Mode **b** is probably a good compromise. However, this mode can cause large numbers of processes if you have a mailer that takes a long time to deliver a message.

---

**Log Level**

The level of logging can be set for **sendmail**. The default using a standard configuration table is level 9. The levels are as follows:

- 0  
No logging.
- 1  
Major problems only.
- 2  
Message collections and failed deliveries.
- 3  
Successful deliveries.
- 4  
Messages being deferred (due to a host being down, etc.).
- 5  
Normal message queue up.
- 6  
Unusual but benign incidents, for example, trying to process a locked queue file.
- 9  
Log internal queue id to external message ID map pings. This can be useful for tracing a message as it travels between several hosts.
- 12  
Several messages that are basically only of interest when debugging.
- 16  
Verbose information regarding the queue.

---

**File Modes**

A number of files may have several modes. The modes depend on what functionality you want and the level of security you require.

### To Suid or Not to Suid?

**Sendmail** can safely be made setuid to root (but only if its version is 5.61 or later). At the point where it is about to `exec(2)` a mailer, it checks to see if the `userid` is zero; if so, it resets the `userid` and `groupid` to a default (set by the `u` and `g` options). (This can be overridden by setting the `S` flag to the mailer for mailers that are trusted and must be called as root.) However, this causes mail processing to be accounted (using `sa(7)`) to root rather than to the user sending the mail.

### Should My Alias Database Be Writable?

At Berkeley we have the alias database (*/usr/lib/aliases\**) mode 644. While this is not as flexible as if the database were more 666, it avoids potential security problems with a globally writeable database.

The database that **sendmail** actually used is represented by the two files *aliases.dir* and *aliases.pag* (both in */usr/ucblib*). The mode on these files should match the mode on */usr/ucblib/aliases*. If *aliases* is writable and the DBM files (*aliases.dir* and *aliases.pag*) are not, users are unable to reflect their desired changes through to the actual database. However, if *aliases* is read-only and the DBM files are writable, a slightly sophisticated user can arrange to steal mail anyway.

If your DBM files are not writable by the world or you do not have auto-rebuild enabled (with the **D** option), then you must be careful to reconstruct the alias database each time you change the text version:

```
newaliases
```

If you ignore or forget this step, any intended changes are also ignored or forgotten.

## The Configuration File

This section describes the configuration file in detail, including hints on how to write one of your own if you have to.

One point should be made clear immediately: the syntax of the configuration file is designed to be reasonably easy to parse, since this is done every time **sendmail** starts up, rather than easy for a human to read or write. On the “future project” list is a configuration-file compiler.

This section includes an overview of the configuration file, followed by details of the semantics.

---

### The Syntax

The configuration file is organized as a series of lines, each of which begins with a single character defining the semantics for the rest of the line. Lines beginning with a space or a tab are continuation lines (although the semantics are not well defined in many places). Blank lines and lines beginning with a sharp symbol (#) are comments.

### R and S — Rewriting Rules

The core of address parsing are the rewriting rules. These are an ordered production system. **Sendmail** scans through the set of rewriting rules looking for a match on the left hand side (LHS) of the rule. When a rule matches, the address is replaced by the right hand side (RHS) of the rule.

There are several sets of rewriting rules. Some of the rewriting sets are used internally and must have specific semantics. Other rewriting sets do not have specifically assigned semantics, and may be referenced by the mailer definitions or by other rewriting sets.

The syntax of these two commands are:

*Sn*

Sets the current ruleset being collected to *n*. If you begin a ruleset more than once it deletes the old definition.

*Rlhs rhs comments*

The fields must be separated by at least one tab character; there may be embedded spaces in the fields. The *lhs* is a pattern that is applied to the input. If it matches, the input is rewritten to the *rhs*. The comments are ignored.

### **D — Define Macro**

Macros are named with a single character. These may be selected from the entire ASCII set, but user-defined macros should be selected from the set of upper case letters only. Lower case letters and special symbols are used internally.

The syntax for macro definitions is:

*Dxval*

where *x* is the name of the macro and *val* is the value it should have. Macros can be interpolated in most places using the escape sequence *\$x*.

### **C and F — Define Classes**

Classes of words may be defined to match on the left hand side of rewriting rules, where a “word” is a sequence of characters that do not contain characters in the *\$o* macro. For example, a class of all local names for this site might be created so that attempts to send to oneself can be eliminated. These can either be defined directly in the configuration file or read in from another file. Classes may be given names from the set of upper case letters. Lower case letters and special characters are reserved for system use.

The syntax is:

```
Ccword1 word2...
Fcfile
```

The first form defines the class *c* to match any of the named words. It is permissible to split them among multiple lines; for example, the two forms:

```
CHmonet ucbmonet
```

and

```
CHmonet
CHucbmonet
```

are equivalent. The second form reads the elements of the class *c* from the named file.

## M — Define Mailer

Programs and interfaces to mailers are defined in this line. The format is:

```
Mname, {field=value}*
```

where *name* is the name of the mailer (used internally only) and the *field=value* pairs define attributes of the mailer. Fields are:

- Path — The pathname of the mailer
- Flags — Special flags for this mailer
- Sender — A rewriting set for sender addresses
- Recipient — A rewriting set for recipient addresses
- Argv — An argument vector to pass to this mailer
- Eol — The end-of-line string for this mailer
- Maxsize — The maximum message length to this mailer

Only the first character of the field name is checked.

## H — Define Header

The format of the header lines that **sendmail** inserts into the message are defined by the H line. The syntax of this line is:

```
H[?mflags?]{hname: htemplate
```

Continuation lines in this spec are reflected directly into the outgoing message. The *htemplate* is macro expanded before it is inserted into the message. If the *mflags* (surrounded by question marks) are specified, at least one of the specified flags must be stated in the mailer definition for this header to be automatically output. If one of these headers is in the input, it is reflected to the output regardless of these flags.

Some headers have special semantics that are described below.

## O — Set Option

A number of “random” options can be set from a configuration file. Options are represented by single characters. The syntax of this line is:

```
Oovalue
```

This sets option *o* to be *value*. Depending on the option, *value* may be a string, an integer, a boolean (with legal values “t”, “T”, “f”, or “F”; the default is TRUE), or a time interval.

## T — Define Trusted Users

Trusted users are those users who are permitted to override the sender address using the **-f** flag. These typically are “root,” “uucp,” and “network,” but on some users, it may be convenient to extend this list to include other users, perhaps to support a separate UUCP login for each host. The syntax of this line is:

```
Tuser1 user2...
```

There may be more than one of these lines.



## P — Precedence Definitions

Values for the “Precedence :” field may be defined using the **P** control line. The syntax of this field is:

```
Pname=num
```

When the *name* is found in a “Precedence :” field, the message class is set to *num*. Higher numbers mean higher precedence. Numbers less than zero have the special property that error messages will not be returned. The default precedence is zero. For example, our list of precedences is:

```
Pfirst-class=0
Pspecial-delivery=100
Pjunk=-100
```

---

## The Semantics

This section describes the semantics of the configuration file.

### Special Macros, Conditionals

Macros are interpolated using the construct `$x`, where *x* is the name of the macro to be interpolated. In particular, lower case letters are reserved to have special semantics, used to pass information in or out of **sendmail**, and some special characters are reserved to provide conditionals, etc.

You can specify conditionals using the syntax:

```
$?x text1 $! text2 $.
```

This interpolates *text1* if the macro `$x` is set, and *text2* otherwise. The “else” (`$!`) clause may be omitted.

The following macros must be defined to transmit information into **sendmail**:

- **c** — The SMTP entry message
- **j** — The “official” domain name for this site
- **l** — The format of the UNIX from line

- **n** — The name of the daemon (for error messages)
- **o** — The set of “operators” in addresses
- **q** — Default format of sender address

The **\$e** macro is printed out when SMTP starts up. The first word must be the **\$j** macro. The **\$j** macro should be in RFC821 format. The **\$l** and **\$n** macros can be considered constants except under terribly unusual circumstances. The **\$o** macro consists of a list of characters which are considered tokens and which separate tokens when doing parsing. For example, if “@” were in the **\$o** macro, then the input “a@b” would be scanned as three tokens: “a,” “@,” and “b.” Finally, the **\$q** macro specifies how an address should appear in a message when it is defaulted.

For example, on our system these definitions are:

```
De$j sendmail $v ready at $b
DnMAILER-DAEMON
DlFrom $g $d
Do.:%@!^=/
Dqq?x ($x)$.
Dj$H.$D
```

An acceptable alternative for the **\$q** macro is “**\$?x\$x \$.<\$g>**”. These correspond to the following two formats:

```
eric@Berkeley (Eric Allman)
Eric Allman <eric@Berkeley>
```

Some macros are defined by **sendmail** for interpolation into argv’s for mailers or for other contexts. These macros are:

- **a** — The origination date in RFC 822 format
- **b** — The current date in RFC 822 format
- **c** — The hop count
- **d** — The date in UNIX (ctime) format
- **f** — The sender (from) address

- **g** — The sender address relative to the recipient
- **h** — The recipient host
- **i** — The queue ID
- **p** — **sendmail**'s pid
- **r** — Protocol used
- **s** — Sender's host name
- **t** — A numeric representation of the current time
- **u** — The recipient user
- **v** — The version number of **sendmail**
- **w** — The hostname of this site
- **x** — The full name of the sender
- **z** — The home directory of the recipient

Three types of dates can be used. The **\$a** and **\$b** macros are in RFC 822 format; **\$a** is the time as extracted from the "Date:" line of the message (if there was one), and **\$b** is the current date and time (used for postmarks). If no "Date:" line is found in the incoming message, **\$a** is set to the current time also. The **\$d** macro is equivalent to the **\$a** macro in UNIX (ctime) format.

The **\$f** macro is the ID of the sender as originally determined; when mailing to a specific host the **\$g** macro is set to the address of the sender relative to the recipient. For example, if I send to *bollard@matisse* from the machine *ucbarpa*, the **\$f** macro is *eric* and the **\$g** macro is *eric@ucbarpa*.

The **\$x** macro is set to the full name of the sender. This can be determined in several ways. It can be passed as flag to **sendmail**. The second choice is the value of the "Full-name:" line in the header if it exists, and the third choice is the comment field of a "From:" line. If all of these fail, and if the message is being originated locally, the full name is looked up in the */etc/passwd* file.

When sending, the **\$h**, **\$u**, and **\$z** macros get set to the host, user, and home directory (if local) of the recipient. The first two are set from the **\$@** and **\$:** part of the rewriting rules, respectively.

The **\$p** and **\$t** macros are used to create unique strings (for example, for the "Message-Id:" field). The **\$i** macro is set to the queue ID on this host; if put into the timestamp line, it can be extremely useful for tracking messages. The **\$v** macro is set to be the version number of **sendmail**; this is normally put in time-stamp and has been proven extremely useful for debugging. The **\$w** macro is set to the name of this host if it can be determined. The **\$c** field is set to the "hop count," that is, the number of times this message has been processed. This count is determined by the **-h** flag on the command line or by counting the number of time-stamps in the message.

The **\$r** and **\$s** fields are set to the protocol used to communicate with **sendmail** and the sending hostname; these are not supported in the current version.

### Special Classes

The class **\$=w** is set to be the set of all names this host is known by. This can be used to match local hostnames.

## The Left Hand Side

The left hand side of rewriting rules contains a pattern. Normal words are simply matched directly. Meta syntax is introduced using a dollar sign. The meta symbols are:

- **\$\*** Match zero or more tokens
- **\$+** Match one or more tokens
- **\$** Match exactly one token
- **\$=x** Match any token in class *x*
- **\$~x** Match any token not in class *x*

If any of these match, they are assigned to the symbol  $\$n$  for replacement on the right hand side, where *n* is the index in the LHS. For example, if the LHS:

```
$-:$+
```

is applied to the input:

```
UCBARPA:eric
```

the rule matches, and the values passed to the RHS are:

```
$1 UCBARPA
$2 eric
```

## The Right Hand Side

When the left hand side of a rewriting rule matches, the input is deleted and replaced by the right hand side. Tokens are copied directly from the RHS unless they begin with a dollar sign. Meta symbols are:

|              |                                               |
|--------------|-----------------------------------------------|
| $\$n$        | Substitute indefinite token <i>n</i> from LHS |
| $\${name\$}$ | Canonical <i>name</i>                         |
| $\$>n$       | "Call" ruleset <i>n</i>                       |
| $\$#mailer$  | Resolve to <i>mailer</i>                      |
| $\$@host$    | Specify <i>host</i>                           |
| $\$:user$    | Specify <i>user</i>                           |

The `$n` syntax substitutes the corresponding value from a `$+`, `$-`, `$*`, `$=`, or `$~` match on the LHS. It may be used anywhere.

A host name enclosed between `$[` and `$]` is looked up using the `gethostent(3)` routines and replaced by the canonical name. For example, `"${csam$}"` might become `"lbl-csam.arpa"` and `"${[[128.32.130.2]]$}"` would become `"vangogh.berkeley.edu."`

The `$>n` syntax causes the remainder of the line to be substituted as usual and then passed as the argument to ruleset *n*. The final value of ruleset *n* then becomes the substitution for this rule.

The `$#` syntax should only be used in ruleset zero. It causes evaluation of the ruleset to terminate immediately, and signals to **sendmail** that the address has completely resolved. The complete syntax is:

```
$#mailer$@host$:user
```

This specifies the `{mailer, host, user}` 3-tuple necessary to direct the mailer. If the mailer is local, the host part may be omitted. The mailer and host must be a single word, but the user may be multi-part.

A RHS may also be preceded by a `$@` or a `$:` to control evaluation. A `$@` prefix causes the ruleset to return with the remainder of the RHS as the value. A `$:` prefix causes the rule to terminate immediately, but the ruleset to continue; this can be used to avoid continued application of a rule. The prefix is stripped before continuing.

The `$@` and `$:` prefixes may precede a `$>` spec; for example:

```
RS+ $:$>7$1
```

matches anything, passes that to ruleset seven, and continues; the `$:` is necessary to avoid an infinite loop.

Substitution occurs in the order described, that is, parameters from the LHS are substituted, hostnames are canonically substituted, “subroutines” are called, and finally \$#, \$@, and \$: are processed.

### Semantics Of Rewriting Rule Sets

Five rewriting sets have specific semantics. These are related as depicted by Figure F-1:

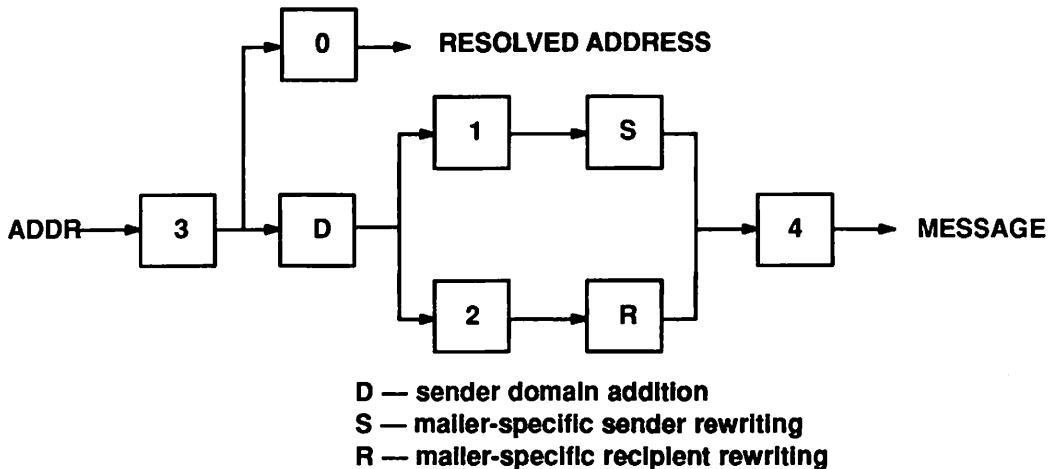


Figure F-1: Rewriting Set Semantics

Ruleset three should turn the address into “canonical form.” This form should have the basic syntax:

```
local-part@host-domain-spec
```

If no “@” sign is specified, then the host-domain spec may be appended from the sender address (if the C flag is set in the mailer definition corresponding to the sending mailer). Ruleset three is applied by **sendmail** before doing anything with any address.

Ruleset zero is applied after ruleset three to addresses that are going to actually specify recipients. It must resolve to a {mailer, host, user} triple. The mailer must be defined in the mailer definitions from the configuration file. The host is defined into the `$h` macro for use in the `argv` expansion of the specified mailer.

Rulesets one and two are applied to all sender and recipient addresses respectively. They are applied before any specification in the mailer definition. They must never resolve.

Ruleset four is applied to all addresses in the message. It is typically used to translate internal to external form.

### Mailer Flags Etc.

A number of flags may be associated with each mailer, each identified by a letter of the alphabet. Many of them are assigned semantics internally. These are detailed in the “Mailer Flags” section. Any other flags may be used freely to conditionally assign headers to messages destined for particular mailers.

```
-“ . nr $1 error” -“ . nr $2 mailer”
```

### The “Error” Mailer

The mailer with the special name “error” can be used to generate a user error. The (optional) host field is a numeric exit status to be returned, and the user field is a message to be printed. For example, the entry:

```
$#error$:Host unknown in this domain
```

on the RHS of a rule causes the specified error to be generated if the LHS matches. This mailer is only functional in ruleset zero.



---

## Building a Configuration File

Building a configuration table can be an extremely difficult job. Fortunately, it is almost never necessary to do so; you can resolve nearly every situation that may come up by changing an existing table. In any case, it is critical that you understand what you are trying to do and come up with a philosophy for the configuration table. This section explains the purpose of a configuration table and gives you some ideas for what your philosophy might be.

### What You Are Trying To Do

The configuration table has three major purposes. The first and simplest is to set up the environment for **sendmail**. This involves setting the options, defining a few critical macros, etc. Since these are described in other places, we will not go into more detail here.

The second purpose is to rewrite addresses in the message. This should typically be done in two phases. The first phase maps addresses in any format into a canonical form. This should be done in ruleset three. The second phase maps this canonical form into the syntax appropriate for the receiving mailer. **Sendmail** does this in three sub-phases. Rulesets one and two are applied to all sender and recipient addresses respectively. After this, you may specify per-mailer rule-sets for both sender and recipient addresses; this allows mailer-specific customization. Finally, ruleset four is applied to do any default conversion to external form.

The third purpose is to map addresses into the actual set of instructions necessary to get the message delivered. Ruleset zero must resolve to the internal form, which is in turn used as a pointer to a mailer descriptor. The mailer descriptor describes the interface requirements of the mailer.

## Philosophy

The particular philosophy you choose depends heavily on the size and structure of your organization. I will present a few possible philosophies here.

One general point applies to all of these philosophies: it is almost always a mistake to try to do full name resolution. For example, if you are trying to get names of the form *user@host* to the Arpanet, it does not pay to route them to *xyzvax!decvax!ucbvax!c70:user@host* since you then depend on several links not under your control. The best approach to this problem is to simply forward to *xyzvax:user@host* and let *xyzvax* worry about it from there. In summary, just get the message closer to the destination, rather than determining the full path.

## Large Site, Many Hosts — Minimum Information

Berkeley is an example of a large site, that is, more than two or three hosts and multiple mail connections. We have decided that the only reasonable philosophy in our environment is to designate one host as the guru for our site. It must be able to resolve any piece of mail it receives. The other sites should have the minimum amount of information they can get away with. In addition, any information they do have should be hints rather than solid information.

For example, a typical site on our local ether network is *monet*. When *monet* receives mail for delivery, it checks whether it knows that the destination host is directly reachable; if so, mail is sent to that host. If it receives mail for any unknown host, it just passes it directly to *ucbvax*, our master host. *Ucbvax* may determine that the host name is illegal and reject the message, or may be able to do delivery. However, it is important to note that when a new mail connection is added, the only host that must have its tables updated is *ucbvax*; the others may be updated if convenient, but this is not critical.

This picture is slightly muddled due to network connections that are not actually located on *ucbvax*. For example, some UUCP connections are currently on *ucbarpa*. However, *monet* does not know about this; the information is hidden totally between *ucbvax* and *ucbarpa*. Mail going from *monet* to a UUCP host is transferred via the ethernet from *monet* to *ucbvax*, then via the ethernet from *ucbvax* to *ucbarpa*, and then is submitted to UUCP. Although this involves some extra hops, we feel this is an acceptable trade-off.

It would be possible to update *monet* to send appropriate UUCP mail directly to *ucbarpa* if the load got too high; if *monet* failed to note a host as connected to *ucbarpa*, it would go via *ucbvax* as before, and if *monet* incorrectly sent a message to *ucbarpa*, it would still be sent by *ucbarpa* to *ucbvax* as before. The only problem that can occur is loops, for example, if *ucbarpa* thought that *ucbvax* had the UUCP connection and vice versa. For this reason, the master host should always be updated first.

This philosophy results as much from the need to have a single source for the configuration files (typically built using *m4(1)* or some similar tool) as any logical need. Maintaining more than three separate tables by hand is essentially an impossible job.

### **Small Site — Complete Information**

A small site (two or three hosts and few external connections) may find it more reasonable to have complete information at each host. This would require that each host know exactly where each network connection is, possibly including the names of each host on that network. As long as the site remains small and the configuration remains relatively static, the update problem will probably not be too great.

## Single Host

This is in some sense the trivial case. The only major issue is trying to ensure that you do not have to know too much about your environment. For example, if you have a UUCP connection, you might find it useful to know about the names of hosts connected directly to you, but this is really not necessary since this may be determined from the syntax.

## Relevant Issues

The canonical form you use should almost certainly be as specified in the Arpanet protocols RFC819 and RFC822. Copies of these RFC's are included on the **sendmail** tape as *doc/rfc819.lpr* and *doc/rfc822.lpr*.

RFC822 describes the format of the mail message itself. **Sendmail** follows this RFC closely, to the extent that many of the standards described in this document cannot be changed without changing the code. In particular, the following characters have special interpretations:

< > ( ) " \

Any attempt to use these characters for other than their RFC822 purpose in addresses is probably doomed to disaster.

RFC819 describes the specifics of the domain based addressing. This is touched on in RFC822 as well. Essentially, each host is given a name which is a right-to-left dot qualified pseudo-path from a distinguished root. The elements of the path need not be physical hosts; the domain is logical rather than physical.

For example, at Berkeley one legal host might be *a.CC.Berkeley.EDU*; reading from right to left, *EDU* is a top level domain comprising educational institutions, *Berkeley* is a logical domain name, *CC* represents the Computer Center, (in this case a strictly logical entity), and *a* is a host in the Computer Center.

Beware when reading RFC819 that there are a number of errors in it.

## **How To Proceed**

Once you have decided on a philosophy, it is worth examining the available configuration tables to decide if any of them are close enough to steal major parts of. Even under the worst of conditions, there is a fair amount of boiler plate that can be collected safely.

The next step is to build ruleset three. This step is the hardest part of the job. Beware of doing too much to the address in this ruleset, since anything you do reflects through to the message. In particular, stripping of local domains is best deferred, since this can leave you with addresses with no domain spec at all. Since **sendmail** likes to append the sending domain to addresses with no domain, this can change the semantics of addresses. Also try to avoid fully qualifying domains in this ruleset. Although technically legal, this can lead to unpleasantly and unnecessarily long addresses reflected into messages. The Berkeley configuration files define ruleset nine to qualify domain names and strip local domains. This is called from ruleset zero to get all addresses into a cleaner form.

Once you have ruleset three finished, the other rulesets should be relatively trivial. If you need hints, examine the supplied configuration tables.

### Testing the Rewriting Rules — The -bt Flag

When you build a configuration table, you can do a certain amount of testing using the “test mode” of sendmail. For example, you could invoke **sendmail** as:

```
sendmail -bt -Ctest.cf
```

which would read the configuration file *test.cf* and enter test mode. In this mode, you enter lines of the form:

```
rwset address
```

where *rwset* is the rewriting set you want to use and *address* is an address to apply the set to. Test mode shows you the steps it takes as it proceeds, finally showing you the address it ends up with. You may use a comma separated list of *rwsets* for sequential application of rules to an input; ruleset three is always applied first.

For example:

```
1,21,4 monet:bollard
```

first applies ruleset three to the input *monet:bollard*. Ruleset one is then applied to the output of ruleset three, followed similarly by rulesets twenty-one and four.

If you need more detail, you can also use the **-d21** flag to turn on more debugging. For example,

```
sendmail -bt -d21.99
```

turns on an incredible amount of information; a single word address is probably going to print out several pages worth of information.

### Building Mailer Descriptions

To add an outgoing mailer to your mail system, you must define the characteristics of the mailer.

Each mailer must have an internal name. This can be arbitrary, except that the names *local* and *prog* must be defined.

The pathname of the mailer must be given in the **P** field. If this mailer should be accessed via an IPC connection, use the string **[IPC]** instead.

The **F** field defines the mailer flags.

- You should specify an **f** or **r** flag to pass the name of the sender as an **-f** or **-r** flag respectively. These flags are only passed if they were passed to **sendmail**, so that mailers that give errors under some circumstances can be placated. If the mailer is not picky, you can just specify **-f \$g** in the argv template.
- If the mailer must be called as root, specify the **S** flag; this does not reset the userid before calling the mailer. (Note that **sendmail** must be running setuid to root for this to work).
- If this mailer is local (that is, performs final delivery rather than another network hop), specify the **l** flag.
- To strip quote characters (backslashes and " marks) from addresses, specify the **s** flag; if it is not given, they are passed through.
- If the mailer can send to more than one user on the same host in a single transaction, specify the **m** flag. If this flag is on, then the argv template containing **\$u** is repeated for each unique user on a given host.
- The **e** flag marks the mailer as being "expensive," which causes **sendmail** to defer connection until a queue run. (You must specify the **c** configuration option for this to be effective.)

An unusual case is the **C** flag. This flag applies to the mailer that the message is received from, rather than the mailer being sent to; if set, the domain spec of the sender (that is, the *@host.domain* part) is saved and is appended to any addresses in the message that do not already contain a domain spec.

For example, a message of the form:

```
From: eric@ucbarpa
To: wnj@monet, mckusick
```

is modified to:

```
From: eric@ucbarpa
To: wnj@monet, mckusick@ucbarpa
```

if and only if the **C** flag is defined in the mailer corresponding to *eric@ucbarpa*.

Other flags are described in the “Mailer Flags” section.

The **S** and **R** fields in the mailer description are per-mailer rewriting sets to be applied to sender and recipient addresses respectively. These are applied after the sending domain is appended and the general rewriting sets (numbers one and two) are applied, but before the output rewrite (ruleset four) is applied. A typical use is to append the current domain to addresses that do not already have a domain.

For example, a header of the form:

```
From: eric
```

might be changed to be:

```
From: eric@ucbarpa
```

or

```
From: ucbvax!eric
```

depending on the domain it is being shipped into. These sets can also be used to do special purpose output rewriting in cooperation with ruleset four.



The **E** field defines the string to use as an end-of-line indication. A string containing only newline is the default. The usual backslash escapes (`\r`, `\n`, `\f`, `\b`) may be used.

Finally, an argv template is given as the **E** field. It may have embedded spaces. If there is no argv with a `$u` macro in it, **sendmail** speaks SMTP to the mailer. If the pathname for this mailer is **[IPC]**, the argv should be

```
IPC $h [port]
```

where *port* is the optional port number to connect to.

For example, the specifications:

```
Mlocal, P=/bin/mail, F=rlsm S=10, R=20, A=mail -d $u
Mether, P=[IPC], F=meC, S=11, R=21, A=IPC $h,
M=100000
```

specify a mailer to do local delivery and a mailer for ethernet delivery. The first is called *local*, is located in the file */bin/mail*, takes a picky **-r** flag, does local delivery, quotes should be stripped from addresses, and multiple users can be delivered at once; ruleset *ten* should be applied to sender addresses in the message and ruleset *twenty* should be applied to recipient addresses; the argv to send to a message is the word *mail*, the word *-d*, and words containing the name of the receiving user. If a **-r** flag is inserted, it will be between the words *mail* and *-d*.

The second mailer is called *ether*; it should be connected to via an IPC connection; it can handle multiple users at once; connections should be deferred; and any domain from the sender address should be appended to any receiver name without a domain; sender addresses should be processed by ruleset *eleven* and recipient addresses by ruleset *twenty-one*. There is a 100,000 byte limit on messages passed through this mailer.

## Command Line Flags

Arguments must be presented with flags before addresses. The flags are:

**-f *addr***

The sender's machine address is *addr*. This flag is ignored unless the real user is listed as a "trusted user" or if *addr* contains an exclamation point (because of certain restrictions in UUCP).

**-r *addr***

An obsolete form of -f.

**-h *cnt***

Sets the "hop count" to *cnt*. This represents the number of times this message has been processed by **sendmail** (to the extent that it is supported by the underlying networks). *Cnt* is incremented during processing, and if it reaches **MAXHOP** (currently 30) **sendmail** throws away the message with an error.

**-F*name***

Sets the full name of this user to *name*.

**-n**

Don't do aliasing or forwarding.

**-t**

Read the header for "To :", "Cc :", and "Bcc :" lines, and send to everyone listed in those lists. The "Bcc :" line is deleted before sending. Any addresses in the argument vector are deleted from the send list.

**-bx**

Set operation mode to *x*. Operation modes are:

```

m - Deliver mail (default)
a - Run in arpanet mode (see below)
s - Speak SMTP on input side
d - Run as a daemon
t - Run in test mode
v - Just verify addresses, don't collect or
 deliver
i - Initialize the alias database
p - Print the mail queue
z - Freeze the configuration file

```

The special processing for the ARPANET includes reading the “From:” line from the header to find the sender, printing ARPANET style messages (preceded by three digit reply codes for compatibility with the FTP protocol [Neigus73, Postel74, Postel77]), and ending lines of error messages with <CRLF>.

**-qtime**

Try to process the queued up mail. If the *time* is given, a **sendmail** runs through the queue at the specified interval to deliver queued mail; otherwise, it only runs once.

**-Cfile**

Use a different configuration file. **Sendmail** runs as the invoking user (rather than root) when this flag is specified.

**-dlevel**

Set debugging level.

**-oxvalue**

Set option *x* to the specified *value*. These options are described in the “Configuration Options” section.

A number of options may be specified as primitive flags (provided for compatibility with deliver mail). These are the *e*, *i*, *m*, and *v* options. Also, the *f* option may be specified as the *-s* flag.

## Configuration Options

The following options may be set using the `-o` flag on the command line or the `O` line in the configuration file. Many of them cannot be specified unless the invoking user is trusted.

### **Afile**

Use the named *file* as the alias file. If no file is specified, use aliases in the current directory.

### **aN**

If set, wait up to *N* minutes for an “@: @” entry to exist in the alias database before starting up. If it does not appear in *N* minutes, rebuild the database (if the `D` option is also set) or issue a warning.

### **Bc**

Set the blank substitution character to *c*. Unquoted spaces in addresses are replaced by this character.

### **c**

If an outgoing mailer is marked as being expensive, do not connect immediately. This requires that queueing be compiled in, since it depends on a queue run process to actually send the mail.

### **dx**

Deliver in mode *x*. Legal modes are:

- `i` - Deliver interactively (synchronously)
- `b` - Deliver in background (asynchronously)
- `q` - Just queue the message (deliver during queue run)

**D**

If set, rebuild the alias database if necessary and possible. If this option is not set, **sendmail** never rebuilds the alias database unless explicitly requested using **-bi**.

**ex**

Dispose of errors using mode *x*. The values for *x* are:

- **p** — Print error messages (default)
- **q** — No messages, just give exit status
- **m** — Mail back errors
- **w** — Write back errors (mail if user not logged in)
- **e** — Mail back errors and give zero exit stat always

**f**

Save Unix-style “From” lines at the front of headers. Normally they are assumed redundant and discarded.

**gn**

Set the default group ID for mailers to run in to *n*.

**Hfile**

Specify the help file for SMTP.

**I**

Insist that the BIND (Berkeley Interact Name Domain) name server be running to resolve host names. If this is not set and the name server is not running, the */etc/hosts* file is considered complete. In general, you want to set this option if your */etc/hosts* file does not include all hosts known to you or if you are using the MX (mail forwarding) feature of the BIND name server (since MX handling is known to be buggy). The name server will still be consulted even if this option is not set, but **sendmail** will feel free to resort to reading */etc/hosts* if the name server is not available. Thus, you should never set this option if you do not run the name server.

### **i**

Ignore dots in incoming messages.

### **Ln**

Set the default log level to *n*.

### **Mxvalue**

Set the macro *x* to *value*. This is intended only for use from the command line.

### **m**

Send to me too, even if I am in an alias expansion.

### **Nnetname**

The name of the home network; “ARPA” by default. The argument of an SMTP “HELO” command is checked against “hostname.netname” where hostname is requested from the kernel for the current connection. If they do not match, “Received:” lines are augmented by the name that is determined in this manner so that messages can be traced accurately.

### **o**

Assume that the headers may be in old format, that is, spaces delimit names. This actually turns on an adaptive algorithm: if any recipient address contains a comma, parenthesis, or angle bracket, it is assumed that commas already exist. If this flag is not on, only commas delimit names. Headers are always output with commas between the names.

### **Qdir**

Use the named *dir* as the queue directory.

***qfactor***

Use *factor* as the multiplier in the map function to decide when to just queue up jobs rather than run them. This value is divided by the difference between the current load average and the load average limit (*x* flag) to determine the maximum message priority that will be sent. Defaults to 10000.

***rtime***

Timeout reads after *time* interval.

***Sfile***

Log statistics in the named *file*.

***s***

Be super-safe when running things, that is, always instantiate the queue file, even if you are going to attempt immediate delivery. **Sendmail** always instantiates the queue file before returning control to the client under any circumstances.

***Ttime***

Set the queue timeout to *time*. After this interval, messages that have not been successfully sent are returned to the sender.

***tS,D***

Set the local timezone name to S for standard time and D for daylight time; this is only used under version six.

***un***

Set the default userid for mailers to *n*. Mailers without the S flag in the mailer definition run as this user.

***v***

Run in verbose mode.

***xLA***

When the system load average exceeds *LA*, just queue messages (that is, do not try to send them).

***XLA***

When the system load average exceeds *LA*, refuse incoming SMTP connections.

***yfact***

The indicated factor is added to the priority (thus lowering the priority of the job) for each recipient, that is, this value penalizes jobs with large numbers of recipients.

***Y***

If set, deliver each job that is run from the queue in a separate process. Use this option if you are short of memory, since the default tends to consume considerable amounts of memory while the queue is being processed.

***zfact***

The indicated factor is multiplied by the message class (determined by the Precedence: field in the user header and the P lines in the configuration file) and subtracted from the priority. Thus, messages with a higher Priority: are favored.

***Zfact***

The factor is added to the priority every time a job is processed. Thus, each time a job is processed, its priority is decreased by the indicated value. In most environments this should be positive, since hosts that are down are all too often down for a long time.



## Mailer Flags

The following flags may be set in the mailer description.

**f**

The mailer wants a **-f** from flag, but only if this is a network forward operation (that is, the mailer gives an error if the executing user does not have special permissions).

**r**

Same as **f**, but sends a **-r** flag.

**S**

Do not reset the `userid` before calling the mailer. This would be used in a secure environment where **sendmail** ran as root. This could be used to avoid forged addresses. This flag is suppressed if given from an “unsafe” environment (for example, a user’s *.mailcf* file).

**n**

Do not insert a UNIX-style “From” line on the front of the message.

**l**

This mailer is local (that is, final delivery will be performed).

**s**

Strip quote characters off of the address before calling the mailer.

**m**

This mailer can send to multiple users on the same host in one transaction. When a **\$u** macro occurs in the `argv` part of the mailer definition, that field is repeated as necessary for all qualifying users.

## Sendmail Installation and Operation

**F**

This mailer wants a "From:" header line.

**D**

This mailer wants a "Date:" header line.

**M**

This mailer wants a "Message-Id:" header line.

**x**

This mailer wants a "Full-Name:" header line.

**P**

This mailer wants a "Return-Path:" header line.

**u**

Upper case should be preserved in user names for this mailer.

**h**

Upper case should be preserved in host names for this mailer.

**A**

This is an Arpanet-compatible mailer, and all appropriate modes should be set.

**U**

This mailer wants Unix-style "From" lines with the ugly UUCP-style "remote from <host>" on the end.

**e**

This mailer is expensive to connect to, so try to avoid connecting normally; any necessary connection will occur during a queue run.

**X**

This mailer wants to use the hidden dot algorithm as specified in RFC821; basically, any line beginning with a dot has an extra dot prepended (to be stripped at the other end). This ensures that lines in the message containing a dot do not terminate the message prematurely.

**L**

Limit the line lengths as specified in RFC821.

**P**

Use the return-path in the SMTP "MAIL FROM:" command rather than just the return address; although this is required in RFC821, many hosts do not process return paths properly.

**I**

This mailer speaks SMTP to another **sendmail**—assuchit can use special protocol features. This option is not required (that is, if this option is omitted, the transmission still operates successfully, although perhaps not as efficiently as possible).

**C**

If mail is received from a mailer with this flag set, any addresses in the header that do not have an at sign (@) after being rewritten by ruleset three has the *@domain* clause from the sender tacked on. This allows mail with headers of the form:

```
From: usera@hosta
To: userb@hostb, userc
```

to be rewritten automatically as:

```
From: usera@hosta
To: userb@hostb, userc@hosta
```

**E**

Escape lines beginning with "From" in the message with a '>' sign.

## Other Configuration

You can make some configuration changes by recompiling **sendmail**. These changes are located in two places:

### *src/conf.h*

Configuration parameters that may be tweaked by the installer are included in *conf.h*.

### *src/conf.c*

Some special routines and a few variables may be defined in *conf.c*. For the most part, these are selected from the settings in *conf.h*.

---

### Parameters In *src/conf.h*

Parameters and compilation options are defined in *conf.h*. Most of these need not normally be tweaked; common parameters are all in *sendmail.cf*. However, the sizes of certain primitive vectors, etc., are included in this file. The numbers following the parameters are their default value.

#### MAXLINE [1024]

The maximum line length of any input line. If message lines exceed this length they are still processed correctly; however, header lines, configuration file lines, alias lines, etc., must fit within this limit.

#### MAXNAME [256]

The maximum length of any name, such as a host or a user name.

#### MAXFIELD [2500]

The maximum total length of any header field, including continuation lines.

**MAXPV [40]**

The maximum number of parameters to any mailer. This limits the number of recipients that may be passed in one transaction.

**MAXHOP [17]**

When a message has been processed more than this number of times, **sendmail** rejects the message on the assumption that there has been an aliasing loop. This can be determined from the **-h** flag or by counting the number of trace fields (that is, "Received:" lines) in the message header.

**MAXATOM [100]**

The maximum number of atoms (tokens) in a single address. For example, the address *eric@Berkeley* is three atoms.

**MAXMAILERS [25]**

The maximum number of mailers that may be defined in the configuration file.

**MAXRWSETS [30]**

The maximum number of rewriting sets that may be defined.

**MAXPRIORITIES [25]**

The maximum number of values for the "Precedence:" field that may be defined (using the P line in *sendmail.cf*).

**MAXTRUST [30]**

The maximum number of trusted users that may be defined (using the T line in *sendmail.cf*).

**MAXUSERENVIRON [40]**

The maximum number of items in the user environment that is passed to subordinate mailers.

**QUEUESIZE [600]**

The maximum number of entries that is processed in a single queue run. A number of other compilation options exist. These specify whether or not specific code should be compiled in.

**DBM**

If set, the DBM package in UNIX is used (see **dbm(3X)** in [UNIX80]). If not set, a much less efficient algorithm for processing aliases is used.

**NDBM**

If set, the new version of the DBM library that allows multiple databases is used. DBM must also be set.

**DEBUG**

If set, debugging information is compiled in. To actually get the debugging output, the **-d** flag must be used.

**LOG**

If set, the syslog routine in use at some sites is used. This makes an informational log record for each message processed, and makes a higher priority log record for internal system errors.

**QUEUE**

This flag should be set to compile in the queueing code. If this is not set, mailers must accept the mail immediately or it is returned to the sender.

**SMTP**

If set, the code to handle user and server SMTP is compiled in. This is only necessary if your machine has some mailer that speaks SMTP.

**DAEMON**

If set, code to run a daemon is compiled in. This code is for 4.2 or 4.3BSD.

**UGLYUUCP**

If you have a UUCP host adjacent to you which is not running a reasonable version of rmail, you must set this flag to include the “remote from sysname” info on the from line. Otherwise, UUCP gets confused about the origin of the mail.

**NOTUNIX**

If you are using a non-UNIX mail format, you can set this flag to turn off special processing of UNIX-style “From” lines.

**NAMED\_BIND**

Compile in code to use the Berkeley Internet Name Domain (BIND) server to resolve TCP/IP host names.

**SETPROCTITLE**

If defined, **sendmail** changes its argv array to indicate its current status. This can be used in conjunction with the **ps** command to find out just what it is up to.

**NO\_WILDCARD\_MX**

Should be set if there are no wildcard MX nameserver records in the local domain. If set, this enables the use of ANY query types, resulting in better performance. Unfortunately, wildcard MX records in the local domain mess this up, hence the need for this compilation option.

---

## **Configuration In src/conf.c**

Not all header semantics are defined in the configuration file. Header lines that should only be included by certain mailers (as well as other more obscure semantics) must be specified in the *HdrInfo* table in *conf.c*. This table contains the header name (which should be in all lower case) and a set of header control flags (described below). The flags are:

**H\_ACHECK**

Normally when the check is made to see if a header line is compatible with a mailer, **sendmail** does not delete an existing line. If this flag is set, **sendmail** deletes even existing header lines. That is, if this bit is set and the mailer does not have flag bits set that intersect with the required mailer flags in the header definition in *sendmail.cf*, the header line is always deleted.

### H\_EOH

If this header field is set, treat it like a blank line, that is, it signals the end of the header and the beginning of the message text.

### H\_FORCE

Add this header entry even if one existed in the message before. If a header entry does not have this bit set, **sendmail** does not add another header line if a header line of this name already existed. This would normally be used to stamp the message by everyone who handled it.

### H\_TRACE

If set, this is a timestamp (trace) field. If the number of trace fields in a message exceeds a preset amount, the message is returned on the assumption that it has an aliasing loop.

### H\_RCPT

If set, this field contains recipient addresses. This is used by the **-t** flag to determine who to send to when it is collecting recipients from the message.

### H\_FROM

This flag indicates that this field specifies a sender. The order of these fields in the HdrInfo table specifies **sendmail**'s preference for which field to return error messages to.



Let's look at a sample *HdrInfo* specification:

```
struct hdrinfo HdrInfo[] =
{
 /* originator fields, most to least significant */
 "resent-sender", H_FROM,
 "resent-from", H_FROM,
 "sender", H_FROM,
 "from", H_FROM,
 "full-name", H_ACHECK,
 /* destination fields */
 "to", H_RCPT,
 "resent-to", H_RCPT,
 "cc", H_RCPT,
 /* message identification and control */
 "message", H_EOH,
 "text", H_EOH,
 /* trace fields */
 "received", H_TRACE|H_FORCE,

 NULL, 0,
};
```

This structure indicates that the "To:", "Resent-To:", and "Cc:" fields all specify recipient addresses. Any "Full Name:" field is deleted unless the required mailer flag (indicated in the configuration file) is specified. The "Message:" and "Text:" fields terminate the header; these are specified in new protocols [NBS80] or used by random dissenters around the network world. The "Received:" field is always added, and can be used to trace messages.

There are a number of important points here. First, header fields are not added automatically just because they are in the *HdrInfo* structure; they must be specified in the configuration file in order to be added to the message. Any header fields mentioned in the configuration file but not mentioned in the *HdrInfo* structure have default processing performed; that is, they are added unless they were in the message already.

Second, the *HdrInfo* structure only specifies cliche'd processing; certain headers are processed specially by ad hoc code regardless of the status specified in *HdrInfo*. For example, the "Sender:" and "From:" fields are always scanned on ARPANET mail to determine the sender; this is used to perform the "return to sender" function. The "From:" and "Full-Name:" fields are used to determine the full name of the sender if possible; this is stored in the macro \$x and used in a number of ways.

The file *conf.c* also contains the specification of ARPANET reply codes. There are four classifications these fall into:

```
char Arpa_Info[] = "050"; /* arbitrary info */
char Arpa_TSyserr[] = "455"; /* some (transient) system error */
char Arpa_PSyserr[] = "554"; /* some (permanent) system error */
char Arpa_Usrerr[] = "554"; /* some (fatal) user error */
```

The class *Arpa\_Info* is for any information that is not required by the protocol, such as forwarding information. *Arpa\_TSyserr* and *Arpa\_PSyserr* are printed by the *syserr* routine. *TSyserr* is printed out for transient errors, that is, errors that are likely to go away without explicit action on the part of a systems administrator. *PSyserr* is printed for permanent errors. The distinction is made based on the value of *errno*. Finally, *Arpa\_Usrerr* is the result of a user error and is generated by the *usrerr* routine; these are generated when the user has specified something wrong, and hence the error is permanent, that is, it does not work simply by resubmitting the request.

If it is necessary to restrict mail through a relay, the *checkcompat* routine can be modified. This routine is called for every recipient address. It can return TRUE to indicate that the address is acceptable and mail processing continues, or it can return FALSE to reject the recipient. If it returns false, it is up to *checkcompat* to print an error message (using *usrerr*) saying why the message is rejected. For example, *checkcompat* could read:

```
bool
checkcompat(to)
 register ADDRESS *to;
{
 if (MsgSize > 50000 && to->q_mailer !=
 LocalMailer)
 {
 usrerr("Message too large for non-local
 delivery");
 NoReturn = TRUE;
 return (FALSE);
 }
 return (TRUE);
}
```

This would reject messages greater than 50000 bytes unless they were local. The **NoReturn** flag can be sent to suppress the return of the actual body of the message in the error return. The actual use of this routine is highly dependent on the implementation, and use should be limited.

---

## Configuration In src/daemon.c

The file *src/daemon.c* contains a number of routines that are dependent on the local networking environment. The version supplied is specific to 4.3 BSD.

The routine *maphostname* is called to convert strings within *[\$ ...]* symbols. You can modify it if you wish to provide a more sophisticated service, for example, mapping UUCP host names to full paths.

## Summary of Support Files

This section summarizes the support files that **sendmail** creates or generates.

*/usr/ucblib/sendmail*

The binary of **sendmail**.

*/usr/ucblib/newaliases*

A link to */usr/ucblib/sendmail*; causes the alias database to be rebuilt. Running this program is equivalent to giving **sendmail** the **-bi** flag.

*/usr/ucblib/mailq*

Prints a listing of the mail queue. This program is equivalent to using the **-bp** flag to **sendmail**.

*/usr/ucblib/sendmail.cf*

The configuration file, in textual form.

*/usr/ucblib/sendmail.fc*

The configuration file represented as a memory image (a.k.a. “frozen”).

*/usr/ucblib/sendmail.hf*

The SMTP help file.

*/usr/ucblib/sendmail.st*

A statistics file; need not be present.

*/usr/ucblib/aliases*

The textual version of the alias file.

*/usr/ucblib/aliases.{pag,dir}*

The alias file in **dbm(3)** format.

*/usr/ucblib/mqueue*

The directory in which the mail queue and temporary files reside.

*/usr/ucblib/mqueue/qf\**

Control (queuec) files for messages.

*/usr/ucblib/mqueue/df\**

Data files.

*/usr/ucblib/mqueue/lf\**

Lock files

*/usr/ucblib/mqueue/tf\**

Temporary versions of the qf files, used during queuec file rebuild.

*/usr/ucblib/mqueue/nf\**

A file used when creating a unique id.

*/usr/ucblib/mqueue/xf\**

A transcript of the current session.

## Error/Status Messages

This section documents **sendmail** error and status messages.

211

System status or system help reply

214

Help message [Information on how to use the receiver or the meaning of a particular non-standard command; this reply is useful only to the human user.]

220

<domain> Service ready

221

<domain> Service closing transmission channel

250

Requested mail action okay, completed

251

User not local; will forward to <forward\_path>

354

Start mail input; end with <CRLF>.<CRLF>

421

<domain> Service not available, closing transmission channel [This may be a reply to any command if the service knows it must shut down.]

450

Requested mail action not taken; mailbox unavailable [for example, mailbox busy]

- 451  
Requested action aborted; local error in processing
- 452  
Requested action not taken; insufficient system storage
- 500  
Syntax error, command unrecognized [This may include errors such as command line too long.]
- 501  
Syntax error in parameters or arguments
- 502  
Command not implemented
- 503  
Bad sequence of commands
- 504  
Command parameter not implemented
- 550  
Requested action not taken; mailbox unavailable [for example, mailbox not found, no access.]
- 551  
User not local; please try <forward\_path>
- 552  
Requested mail action aborted; exceeded storage allocation
- 553  
Requested action not taken; mailbox name not allowed [for example, mailbox syntax incorrect]
- 554  
Transaction failed

# Glossary

## **Active System**

A system with Basic Networking Utilities and the hardware required to establish communication links (i.e. Auto-Dial Modem).

## **Address**

A number, label, or name that indicates the location of information in the computer's memory.

## **A.out**

The default name of a freshly compiled object file. Historically a.out signifies assembler output.

## **Archive**

(1) A collection of data gathered from several files into one file;  
(2) especially, such a collection gathered by **ar**(1) for use as a library.

## **Automatic calling unit**

A hardware device used to dial stored telephone numbers. It allows the system to contact another system over phone lines without manual intervention.

## **Bad block**

A section of a storage medium that cannot store data reliably.

## **Basic Networking Utilities (BNU)**

A group of programs and files that permit copy capability between UNIX Systems. Sometimes called UUCP.

## **Block**

The basic unit of buffering in the kernel. See indirect, logical, and physical blocks.



**Block device**

A device upon which a file system can be mounted; typically a permanent storage device such as a tape or disk drive, so called because data transfers to the device occur by blocks.

**Boot**

To start the operating system, so called because the kernel must bootstrap itself from secondary storage into an empty machine. No login or process persists across a boot.

**Boot block**

The first block of a file system that is reserved for a booting program.

**Boot program**

Loads the operating system into memory.

**Buffer**

(1) A staging area for input/output where arbitrary-length transactions are collected into convenient units for system operations. (2) To use buffers.

**Buffer pool**

A region of storage available to the file system for holding blocks. All input/output for block devices goes through the buffer pool so read and write operations may be independent of device blocks.

**Cartridge tape**

A storage medium that consists of a magnetic tape wound on spools housed in a plastic container.

**Character device**

A device upon which a file system cannot be mounted such as a terminal or the null device.

**Child process**

See fork.

**Controller**

A device that directs the transmission of data over the data links of a network.

**Crash**

If a hardware or software error condition develops that the system can not handle, it takes itself out of service, or crashes. Such conditions occur when the system can not allocate resources, manage processes, respond to requests for system functions, or when the electrical power is unstable.

**Cron**

A command which creates a daemon that invokes commands at specified dates and times.

**Daemon**

A background process, often perpetual, that performs a system-wide public function, e.g. **calendar**(1) and **cron**(8).

**Device**

(1) A special file such as a tape drive or the nulldevice; not a regular file or directory. (2) A physical input/output unit.

**Directory hierarchy**

The tree of all directories, in which each is reachable from the root via a chain of subdirectories.

**Disk**

A platter coated with magnetic material on which data can be stored.

**Diskette, flex disk**

A magnetic storage medium that is smaller and originally more flexible than a hard disk.

**Drive**

The hardware device that holds magnetic disks, diskettes, and tapes while they are in use.

**Dump**

A copy of the memory of the system, used for analyzing problems.

**Executable file**

(1) An object file that is ready to be copied into the address space of a process to run as the code of that process. (2) A file that has execute permission, either an executable file or a shell script.

**File system**

(1) A collection of files that can be mounted on a block special file; each file of a file system is accessible via some path from the root directory of the file system. (2) The collection of all files on a computer.

**Filter**

A program that reads from the standard input and writes on the standard output, so called because it can be used as a data-transformer in a pipeline.

**Flush**

To empty a buffer.

**Fork**

To split one process into two (the parent process and child process) with separate, but initially identical, text, data, and stack segments.

**Free list**

In a file system, the list of blocks that are not occupied by data.

**Getty**

One of a series of processes which may connect the user to the UNIX system. In NCR UNIX System V Release 4, the **ttymon** (and **sac**) processes are normally used instead of **getty**. If used, **getty** is invoked by **init**, and in turn invokes **login**.

### **Group**

(1) A set of permissions alternative to owner permissions for access to a file. (2) A set of user IDs that may assume the privileges of a group. (3) The group ID of a file.

### **Group ID**

An integer value, usually associated with one or more login names; as the user ID of a process becomes the owner of files created by the process, so the group ID of a process becomes the group of such files.

### **Hole**

A gap in a file caused by seeking while writing. **read(2)** takes data in holes to be zero. A block in a hole occupies no space in its file system. Files with holes in them are known as sparse files.

### **I-list**

The index to a file system listing all the inodes of the file system.

### **Indirect blocks**

Data blocks that are not directly referenced by an inode. The inode has addresses that indirectly reference (by a cascade of pointers) an additional number of blocks to handle extremely large file sizes.

### **Init**

A general process spawner that is invoked as the last step in the **boot** procedure; it regularly checks a table that defines what processes should run at what run level.

### **Inode**

An element of a file system; an inode specifies all properties of a particular file and locates the file's contents, if any.

### **Inode number, I-number**

The position of an inode in the I-list of a file system.

### **Interface programs**

Shell scripts (for example, those furnished with the LP Spooler software) that interface between the user and some part of the overall system (for example, the printer).

### **Kernel**

The UNIX system proper; resident code that implements the system calls.

### **Kernel address space**

A portion of memory used for data and memory addressable only by the kernel.

### **Link**

(1) To add an entry for an existing file to a directory; converse of unlink. (2) By extension, a directory entry.

### **Link count**

The number of directory entries that pertain to an inode; a file ceases to exist when its link count becomes zero and it is not open.

### **Load device**

Designates the physical device from which a program will be loaded into main memory.

### **Local system**

Refers to the system on the near end of a communication link; normally, your system.

### **Log files**

Contain records of transactions that occur on the system; software that spools, for example, generates various log files.

### **Logical block**

A unit of data as it is handled by the software. The system handles data in logical blocks whose size is determined by the file system.

**Memory image**

A copy of all the segments of a running or terminated program. The copy may exist in main storage, in the swap area, or in a file.

**Mode, file mode**

The permissions of a file; colloquially referred to by a 3-digit octal number, e.g. 'a 755 file'. See **chmod(1)**.

**Mount**

To extend the directory hierarchy by associating the root of a file system with a directory entry in an already mounted file system. The converse is unmount, spelled **umount**.

**Namelist**

Same as symbol table.

**Network**

The hardware and software that constitute the interconnections between computer systems, permitting electronic communication between the systems and associated peripherals.

**Networking**

For computer systems, this means sending data from one system to another over some communications medium (coaxial cable, phone lines, etc.). Common networking services include file transfer, remote login, remote execution.

**Node**

A terminating point (system) on a network.

**Node name**

The name for the system (may be up to 255 characters); used as the official name of the machine in a network. The node name resides in the **NODE** parameter.

**Null device**

A device that always yields end-of-file on reading and discards all data on writing.

### **Object file**

A file of machine language code and data; object files are produced from source programs by compilers and from other object files and libraries by the link editor. An object file that is ready to run is an executable file.

### **Operating system**

The program for managing the resources of the computer. It takes care of such things as input/output procedures, process scheduling, the file system, removing this burden from user programs.

### **Open file**

(1) The destination for input or output obtained by opening a file or creating a pipe; a file descriptor. Open files are shared across forks and persist across executes. (2) Loosely, a file that has been opened. However an open file need not exist in a file system, and a file may be the destination of several open files simultaneously.

### **Other**

A set of permissions regulating access to a file by processes with userID different from the owner and groupID different from the group of the file.

### **Owner**

The userID of the process that created a file; the owner has distinctive permissions for a file.

### **Page**

A fixed length, a 4096-byte portion of memory that has a virtual address, and that can be transferred between main and secondary storage.

### **Paging**

The process by which programs are truncated into pages and transferred between main and secondary storage by the virtual handler (or paging daemon).

**Parent process**

See **fork**.

**Partitions**

Units of storage space on disk.

**Passive system**

A system that has Basic Networking Utilities but does not have the hardware required to establish communication links and so never initiates calls.

**Permission**

A right to access a file in a particular way including read, write, execute (or look up in, if a directory). Permissions are granted separately to owner, group, and others.

**Permission bit**

A permission, so called because each permission is encoded into one bit in an inode.

**Physical block**

A unit of data as it is actually stored and manipulated. The system handles data in 512-byte physical blocks.

**Physical memory**

See **memory**.

**Pipe**

A direct stream connection between processes, whereby data written on an open file in one process becomes available for reading in another.

**Pipeline**

A sequence of programs connected by pipes.

**Polling**

The interrogation of devices by the operating system to avoid contention, determine operation status, or ascertain readiness to send or receive data.



**Ports**

The point of physical connection between a peripheral device (such as a terminal or a printer) and the device controller, that is part of the computer hardware.

**Process**

A connected sequence of computation. A process is characterized by a core image with instruction location counter, current directory, a set of open files, control terminal, userid, and groupid.

**Process id**

An integer that identifies a process.

**Process number**

Same as process ID.

**Profile**

(1) An optional shell script, .profile, conventionally used by the shell upon logging in to establish the environment and other working conditions customary to a particular user. (2) To collect a histogram of values of the instruction location counter of a process.

**Program**

(1) An executable file. (2) A process. (3) All the usual meanings.

**Queue**

A line or list formed by items in a system waiting for service.

**Raw device**

A block device, read and write operations to which are not buffered, and are synchronized to natural records of the physical device.

**Reboot**

Same as boot.

**Remote system**

Refers to a system on the far end of a communication link; normally, a system that your system calls.

**Retension**

The process of rewinding the tape in a cartridge tape device to make sure it is at the correct tautness for accurate recording of data.

**Root**

(1) A distinguished directory that constitutes the origin of the directory hierarchy in a file system. (2) Specifically, the origin for the file system, with the conventional path name “/”.  
(3) The origin of the directory hierarchy in a file system.

**Run level**

A software configuration of the system which allows a particular group of processes to exist.

**Schedule**

To assign resources – main store and CPU time – to processes.

**Scheduler**

A permanent process, and associated kernel facilities that does swapping.

**Search path**

In the shell, a list of path names of directories that determines the meaning of a command; the command name is prefixed with members of the search path in turn until a path name of an executable file results; the search path is given by the shell variable PATH.

**Section, sector**

A 512-byte portion of a track which can be accessed by the magnetic disk heads in the course of a predetermined rotational displacement of the storage device.

### **Segment**

A contiguous range of the address space of a process with consistent storage access capabilities; the four segments are (1) the text segment, occupied by executable code, (2) the data segment, occupied by static data that is specifically initialized, (3) the bss.segment, occupied by static data that is initialed by default to zero values, and (4) the stack segment, occupied by automatic data, see stack. Sometimes (2), (3), and (4) are collectively called data segments.

### **Semaphore**

An IPC facility that allows two or more processes to be synchronized.

### **Set userid**

A special permission for an executable file that causes a process executing it to have the access rights of the owner of the file. The owner's user ID becomes the effective user ID of the process, distinguished from the real user ID under which the process began.

### **Set userid bit**

The associated permission bit.

### **Shared memory**

An IPC facility that permits two or more processes to share the same data space.

### **Shell**

(1) The program `sh(1)`, that causes other programs to be executed on command; the shell is usually started on a user's behalf when the user logs in. (2) By analogy, any program started upon logging in.

### **Shell script**

An executable file of commands taken as input to the shell.

**Single user**

A state of the operating system in which only one user is supported.

**Source file**

(1) The uncompiled version of a program. (2) Generally, the unprocessed version of a file.

**Special file**

An inode that designates a device, further categorized as either (1) a block special file describing a block device, or (2) a character special file describing a character device.

**Spool**

To collect and serialize output from multiple processes competing for a single output service.

**Spooler**

A daemon that spools.

**Startup**

Same as boot.

**Super block**

The second block in a file system, that describes the allocation of space in the file system.

**Superuser**

userid 0 with access to any file regardless of permissions. The superuser can perform certain privileged system calls, e.g. setting the clock.

**Swap**

To move the memory image of an executing program between main and secondary storage to make room for other processes.

**Swap area**

The part of secondary storage to which memory images are swapped. The swap area is disjointed from the file system.

**Symbolic link**

A file that contains the path name of another file or directory. References to the symbolic link become references to the named file or directory.

**Symbol table**

Information in an object file about the names of data and functions in that file; the symbol table and address relocation information are used by the link editor to compile object files and by debuggers.

**System calls**

(1) The set of system primitive functions through which all system operations are allocated, initiated, monitored, manipulated, and terminated. (2) The system primitives invoked by user processes for system-dependent functions, such as I/O, process creation, etc.

**System console**

The directly connected terminal used for communication between the operator and the computer.

**System name**

The name (up to 255 characters) for the system; resides in the SYS parameter. This is the name given by the manufacturer, indicating the version of your operating system. It is NOT the node name, a name that uniquely identifies your system for communication with other systems.

**Text file, ASCII file**

A file, the bytes of which are understood to be in ASCII code.

**Track**

An addressable ring of sections on a disk or flex disk; each disk or flex has a predefined number of concentric tracks, which allows the disk head to properly access sections of data.

**Tunable parameters**

Variables used to set the sizes and thresholds of the various control structures of the operating system.

**Tuning**

(1) Modifying the tunable parameters to improve system performance. (2) The reconfiguration of the operating system to incorporate the modifications into an executable version of the system.

**UUCP**

A group of programs and files that permit UNIX-to-UNIX copy capability between UNIX Systems. If shown in the text with bold type(**uucp**), this is referring specifically to the **uucp(1)** program or login ID. Also, called Basic Networking Utilities (BNU).

**UserID**

An integer value, usually associated with a login name. The userID of a process becomes the owner of files created by the process and descendent (forked) processes.

**Utility, utility program**

A standard, generally useful, permanently available program.

# Index

---

## A

### access

Maintenance File System: *Vol. 1* 2–11

to remote systems: *Vol. 2* 9–35

access the system: *Vol. 1* 1–2

activate, processor: *Vol. 1* 1–33

active partition, change: *Vol. 2* 1–24

### add

disk: *Vol. 2* 1–3

SCSI controller: *Vol. 2* 2–27, 2–34

swap file: *Vol. 2* 1–27

uucp logins: *Vol. 2* 9–20

### adjust

printer port characteristics: *Vol. 2* 6–12

terminfo database: *Vol. 2* 6–14

AFFIN\_ON: *Vol. 3* 3–42, 4–2

AFFINDECAY: *Vol. 3* 3–42, 4–3

affinity scheduling parameters: *Vol. 3* 3–42

AGEINTERVAL: *Vol. 3* 3–32

AIOTIMEOUT: *Vol. 3* 3–40, 4–4

### alert

receive for printer fault: *Vol. 2* 5–8

stop for printer fault: *Vol. 2* 5–9

alias database, sendmail: *Vol. 1* F–15

aliasing, sendmail: *Vol. 1* E–14

alignment pattern: *Vol. 2* 5–14

allow, access to printer: *Vol. 2 4-46*  
alternate boot disk: *Vol. 1 1-17*  
    boot from: *Vol. 1 1-24*  
analyze  
    activity data: *Vol. 3 1-7*  
    specific process with timex: *Vol. 3 1-47*  
    specific processes with time: *Vol. 3 1-45*  
analyze system activity, with process accounting: *Vol. 3 1-35*  
ARCHITECTURE: *Vol. 3 3-47, 4-5*  
ARG\_MAX: *Vol. 3 3-29, 4-6*  
argument, sendmail: *Vol. 1 F-19*  
asynchronous I/O  
    maximum outstanding calls parameter: *Vol. 3 4-69*  
    timeout value parameter: *Vol. 3 4-4*

---

**B**

backing up, entire system: *Vol. 1 6-19*  
backup  
    file system and disk information: *Vol. 1 6-16*  
    file systems: *Vol. 1 6-4*  
    raw devices: *Vol. 1 6-14*  
bad block, mark: *Vol. 2 1-19*  
Basic Networking Utilities, (see BNU): *Vol. 2 8-2*  
baud rate, printer: *Vol. 2 4-51*  
BDFLUSHR: *Vol. 3 3-29*  
bfs file system, check: *Vol. 1 5-78*  
bfs file system type: *Vol. 1 A-23*  
bind, display information for processes: *Vol. 1 1-35*  
bind a process: *Vol. 1 1-35*  
block, mark bad: *Vol. 2 1-19*  
BMAX: *Vol. 3 3-44*



**BNU**

administrative support files: *Vol. 2 8–12*

cleanup undeliverable jobs: *Vol. 2 9–68*

commands: *Vol. 2 8–5*

description of: *Vol. 2 8–2*

maintain: *Vol. 2 9–64*

making physical connection: *Vol. 2 9–7*

set up links: *Vol. 2 9–5*

BNU data base: *Vol. 2 8–10*

BNU link, verify: *Vol. 2 9–63*

BNU log file, check size of: *Vol. 2 9–68, 9–70*

BNU logs: *Vol. 2 8–16*

**boot**

from alternate disk: *Vol. 1 1–17*

from flex diskettes: *Vol. 1 2–4*

kernel: *Vol. 3 3–15*

**boot flex diskettes**

copy: *Vol. 1 2–7*

use: *Vol. 1 2–2*

BOOT program, recover: *Vol. 1 3–29*

buffer, number reserved for pageout demon: *Vol. 3 4–106*

buffer utilization: *Vol. 3 7–4*

BUFHWM: *Vol. 3 3–29, 4–7*

build, sendmail configuration file: *Vol. 1 F–39*

---

**C**

call-out table, maximum entry parameter: *Vol. 3 4–74*

cartridge tape, retension: *Vol. 1 6–26*

cblocks, maximum parameter: *Vol. 3 4–78*

cdfs file system type: *Vol. 1 A–27*

change

active partition: *Vol. 2 1–24*

keyboard mapping: *Vol. 1 D–4*

line characteristics to 8 bit: *Vol. 1 D–2*

menu entry: *Vol. 1 C–18*

run levels: *Vol. 1 1–5*

screen mapping: *Vol. 1 D–6*

character set: *Vol. 2 5–24*

list: *Vol. 2 5–26*

map: *Vol. 2 5–26*

name: *Vol. 2 5–25*

characteristics, adjust for printer port: *Vol. 2 6–12*

check

file system: *Vol. 1 5–4*

quotas: *Vol. 1 4–21*

size of BNU log files: *Vol. 2 9–68, 9–70*

ufs file systems: *Vol. 1 5–38*

check bfs file system: *Vol. 1 5–78*

cleanup

public area: *Vol. 2 9–68, 9–69*

undeliverable BNU jobs: *Vol. 2 9–68*

CMAX: *Vol. 3 3–44*

CMF: *Vol. 3 3–39, 4–8*

collect activity data, with sadc: *Vol. 3 1–5*

COM2CONS: *Vol. 3 3–39*

combine, BNU log files: *Vol. 2 9–68, 9–69*

commands for BNU: *Vol. 2 8–5*

configuration

files and directories: *Vol. 3 3–2*

parameters: *Vol. 3 3–25*

sendmail: *Vol. 1 E–16*

- configuration file
  - build for sendmail: *Vol. 1 F-39*
  - sendmail: *Vol. 1 F-10, F-27*
- configuration options, sendmail: *Vol. 1 F-50*
- Configure, modems: *Vol. 2 3-29*
- configure
  - dump area: *Vol. 2 1-31*
  - LP Print Service: *Vol. 2 4-3*
  - modem on serial controller: *Vol. 2 3-25*
  - printers: *Vol. 2 4-12*
  - second serial port: *Vol. 2 3-20*
  - serial controller ports for terminals: *Vol. 2 3-22*
  - serial port for a modem: *Vol. 2 3-15*
  - serial port for terminal: *Vol. 2 3-13*
  - serial port for three-wire cable: *Vol. 2 3-35*
- control, disk usage: *Vol. 3 7-7*
- copy
  - boot flex diskettes: *Vol. 1 2-7*
  - maintenance and reference diskettes: *Vol. 1 2-6*
  - maintenance flex diskette: *Vol. 1 2-7*
- copy files, to NCR TOWERS: *Vol. 1 6-25*
- copyright, parameter: *Vol. 3 4-80*
- core file, size limit parameter: *Vol. 3 4-122*
- corruption, file system: *Vol. 1 5-2*
- cpu workload: *Vol. 3 8-2*
- crash utility: *Vol. 1 4-6*
- create
  - access mechanisms: *Vol. 2 9-35*
  - file system: *Vol. 1 4-7*
  - kernel: *Vol. 3 3-11*
  - menu entry: *Vol. 1 C-18*
  - security mechanisms: *Vol. 2 9-35*
- cron: *Vol. 2 9-64*

crontab command: *Vol. 2 9–65*  
cu command: *Vol. 2 9–73*  
customize  
    interface program: *Vol. 2 6–22*  
    print service: *Vol. 2 6–12*

---

**D**

daemon mode, sendmail: *Vol. 1 F–19*  
daemons, networking: *Vol. 2 8–8*  
data base for BNU: *Vol. 2 8–10*  
data segment, maximum size parameter: *Vol. 3 4–124*  
daylight savings time, parameter: *Vol. 3 4–12*  
deactivate, processor: *Vol. 1 1–33*  
debug, sendmail: *Vol. 1 F–20*  
default  
    priority setting: *Vol. 2 5–3*  
    set for print queue priority: *Vol. 2 5–3*  
define, filter: *Vol. 2 5–35*  
delete  
    menu: *Vol. 1 C–31*  
    task entry: *Vol. 1 C–31*  
delivery mode, sendmail: *Vol. 1 F–24*  
demand paging: *Vol. 3 6–3*  
deny, access to printer: *Vol. 2 4–46*  
DESFREE: *Vol. 3 3–32, 4–9, 6–5*  
device, protocols: *Vol. 2 9–27*  
device name: *Vol. 1 B–5*  
device numbers, major and minor: *Vol. 1 B–2*  
devices, used for BNU communication: *Vol. 2 9–21*  
Devices file: *Vol. 2 9–21*  
df command: *Vol. 1 4–15; Vol. 3 7–8, 7–10*

- diagnostic diskette: *Vol. 1 2–2*
- dial out failure: *Vol. 2 4–54*
- Dial-up password protection
  - disable: *Vol. 2 3–34*
  - enable: *Vol. 2 3–32*
- Dialers file: *Vol. 2 9–30*
- dialing, international calls: *Vol. 2 9–34*
- dialing information, write: *Vol. 2 9–30*
- Direct Memory Access (DMA)
  - exclusive ownership parameter: *Vol. 3 4–11*
  - maximum click number parameter: *Vol. 3 4–45*
  - pages reserved for: *Vol. 3 4–10*
- directories, reorganizing: *Vol. 3 7–14*
- Disable dial-up password protection: *Vol. 2 3–34*
- disk
  - add: *Vol. 2 1–3*
  - display slicing information: *Vol. 2 1–23*
  - format: *Vol. 2 1–6*
  - monitor and control usage: *Vol. 3 7–7*
  - monitor space used: *Vol. 3 7–8*
  - monitor usage: *Vol. 3 7–7*
  - partition: *Vol. 2 1–8*
  - performance: *Vol. 3 7–2*
  - replace: *Vol. 2 1–35*
  - slice: *Vol. 2 1–14*
- disk space, identify large users: *Vol. 3 7–12*
- disk usage, monitor and control: *Vol. 3 7–7*
- diskadd, use: *Vol. 2 1–2*
- diskadd command: *Vol. 2 1–4*
- diskette
  - diagnostics: *Vol. 1 2–2*
  - reference: *Vol. 1 2–2*

## display

file system information: *Vol. 1 4–15*filter: *Vol. 2 5–44*form: *Vol. 2 5–18*installed file system types: *Vol. 1 4–6*process binding information: *Vol. 1 1–35*queue priority: *Vol. 2 5–3*slicing information: *Vol. 2 1–23*status of printers: *Vol. 2 4–37*status of processors: *Vol. 1 1–33*dkformat command: *Vol. 2 1–7*dklayout command: *Vol. 2 1–13*DMAABLEBUF: *Vol. 3 3–38, 4–10*DMAEXCL: *Vol. 3 3–38, 4–11*DO386B1: *Vol. 3 3–38*DO387CR3: *Vol. 3 3–38*download fonts: *Vol. 2 6–11*DSTFLAG: *Vol. 3 3–38, 4–12*

## dump area

configure: *Vol. 2 1–31*examine contents: *Vol. 1 3–39*save memory to: *Vol. 1 3–35*save to media: *Vol. 1 3–37*

---

**E**edquota command: *Vol. 1 4–21*edsysadm command: *Vol. 1 C–6*Enable dial-up password protection: *Vol. 2 3–32*entire disk, restore: *Vol. 1 7–10*

## entire system

backing up: *Vol. 1 6–19*restoring: *Vol. 1 7–20*

/etc/conf: *Vol. 3* 3–3  
 /etc/d\_passwd: *Vol. 2* 3–32  
 /etc/dialups: *Vol. 2* 3–32  
 /etc/netconfig: *Vol. 2* 7–2  
 /etc/passwd: *Vol. 1* 1–28  
     recover: *Vol. 1* 3–13  
 /etc/shadow: *Vol. 1* 1–28  
     recover: *Vol. 1* 3–17  
     update: *Vol. 1* 1–29  
 /etc/uucp: *Vol. 2* 8–10  
 /etc/vfstab: *Vol. 1* 4–4  
 Ethernet LAN Module, install: *Vol. 2* 2–23  
 EVDATA: *Vol. 3* 3–41  
 EVFNHTS: *Vol. 3* 3–41  
 EVMAXDPE: *Vol. 3* 3–41  
 EVMAXETERMS: *Vol. 3* 3–41  
 EVMAXEV: *Vol. 3* 3–41  
 EVMAXMEM: *Vol. 3* 3–41  
 EVMAXTRAPS: *Vol. 3* 3–41  
 EVTIDHTS: *Vol. 3* 3–41  
 exit codes, printer interface program: *Vol. 2* 6–24

---

**F**

failure, dial out: *Vol. 2* 4–54  
 fault, printer: *Vol. 2* 5–7  
 FDFLUSHR: *Vol. 3* 3–29, 4–13  
 file  
     copy to NCR TOWERS: *Vol. 1* 6–25  
     identifying inactive: *Vol. 3* 7–11  
     removing inactive: *Vol. 3* 7–11  
 file descriptor, maximum per process: *Vol. 3* 4–135

file mode, sendmail: *Vol. 1 F-25*

file system

check and repair: *Vol. 1 1-12, 5-4*

check from maintenance file system: *Vol. 1 3-40*

create: *Vol. 1 4-7*

display information about: *Vol. 1 4-15*

eliminate fragmentation: *Vol. 3 7-16*

identify type of: *Vol. 1 4-6*

integrity: *Vol. 1 5-2*

make: *Vol. 1 4-7*

maximum mounted parameter: *Vol. 3 4-87*

minimizing corruption: *Vol. 1 5-2*

mount: *Vol. 1 4-11*

organization: *Vol. 1 A-2, A-6*

unmount: *Vol. 1 4-13*

file system type

bfs: *Vol. 1 A-23*

cdfs: *Vol. 1 A-27*

list: *Vol. 1 4-6*

parameter for root: *Vol. 3 4-118*

s5: *Vol. 1 A-10*

ufs: *Vol. 1 A-17*

file system types: *Vol. 1 A-8*

file system and disk information, backup: *Vol. 1 6-16*

file transfer protocol: *Vol. 2 9-60*

filter: *Vol. 2 5-32*

change: *Vol. 2 5-44*

define: *Vol. 2 5-35*

display: *Vol. 2 5-44*

fast: *Vol. 2 6-26*

PostScript: *Vol. 2 6-7*

remove: *Vol. 2 5-44*

restore defaults: *Vol. 2 5-45*

FLCKREC: *Vol. 3 3-29, 4-14*



## font

downloading: *Vol. 2 6–11*

PostScript: *Vol. 2 6–8*

force queue, sendmail: *Vol. 1 F–19*

## form

display: *Vol. 2 5–18*

provide to printer: *Vol. 2 5–12*

restrict user access: *Vol. 2 5–16*

format, disk: *Vol. 2 1–6*

format command: *Vol. 2 1–7*

forward files, sendmail: *Vol. 1 F–18*

forwarding, sendmail: *Vol. 1 E–14*

free block list, reorganize: *Vol. 3 5–5*

free memory, minimum parameter: *Vol. 3 4–58*

## fsck

automatic checking: *Vol. 1 1–12, 1–13*

from maintenance file system: *Vol. 1 3–40*

manual checking: *Vol. 1 1–12, 1–14*

on bfs file systems: *Vol. 1 5–78*

phases on ufs file systems: *Vol. 1 5–47*

fstyp command: *Vol. 1 4–6*

---

**G**

GPGSHI: *Vol. 3 3–32*

GPGSLO: *Vol. 3 3–32, 4–15, 6–7*

GPGSMK: *Vol. 3 3–32*

---

**H**

hard disk, display slicing information: *Vol. 2 1–23*

hard limit parameter: *Vol. 3 1–6*

hardware provider, parameter for name: *Vol. 3 4–25*

hash buffer, maximum parameter: *Vol. 3 4–84*

HCORLIM: *Vol. 3* 3-43, 4-16  
HCPULIM: *Vol. 3* 3-43, 4-17  
HDATLIM: *Vol. 3* 3-43, 4-18  
header declarations, sendmail: *Vol. 1* E-17  
help messages, write: *Vol. 1* C-8  
HFNOLIM: *Vol. 3* 3-43, 4-19  
HFSZLIM: *Vol. 3* 3-43  
hold, print request: *Vol. 2* 5-5  
host-resident fonts: *Vol. 2* 6-9  
HRSZLIM: *Vol. 3* 4-20  
HRTIME: *Vol. 3* 3-42, 4-21  
HRVTIME: *Vol. 3* 3-42, 4-22  
HSTKLIM: *Vol. 3* 3-43, 4-23  
HVMMLIM: *Vol. 3* 3-43, 4-24  
HW\_PROVIDER: *Vol. 3* 3-47, 4-25  
HW\_SERIAL: *Vol. 3* 3-47, 4-26

---

I/O balancing: *Vol. 3* 7-3

identify

    devices used for BNU communication: *Vol. 2* 9-21

    inactive file: *Vol. 3* 7-11

    systems for BNU communications: *Vol. 2* 9-11

    type of file system: *Vol. 1* 4-6

idle printer: *Vol. 2* 4-55

id tune command: *Vol. 3* 3-26

ILDMAXOPENS: *Vol. 3* 3-45

illegible output, printers: *Vol. 2* 4-50

inactive file, identify and remove: *Vol. 3* 7-11

inclusion, sendmail: *Vol. 1* E-15

init command: *Vol. 1* 1-5

inode, maximum parameter for s5: *Vol. 3 4–85*

/install: *Vol. 1 2–11*

## install

Ethernet LAN Module: *Vol. 2 2–23*

from remote console: *Vol. 1 1–47*

LP Print Service: *Vol. 2 4–11*

Postscript fonts: *Vol. 2 6–8*

PostScript printer: *Vol. 2 6–7*

PostScript printers: *Vol. 2 6–6*

remote installation equipment requirements: *Vol. 1 1–48*

sendmail: *Vol. 1 F–4*

serial controller: *Vol. 2 2–18*

setting up remote console: *Vol. 1 1–49*

software packages from flex: *Vol. 1 1–30*

software packages from tape: *Vol. 1 1–30*

installation, from remote console: *Vol. 1 1–47*

installation tape, recover files from: *Vol. 1 3–21*

integrity of file system: *Vol. 1 5–2*

interface program: *Vol. 2 6–18*

customize: *Vol. 2 6–22*

international calls, dial: *Vol. 2 9–34*

item help file: *Vol. 1 C–6*

---

## K

kdb symbol table, maximum size parameter: *Vol. 3 4–31*

KDBSYMSIZE: *Vol. 3 3–38, 4–31*

## kernel

create new: *Vol. 3 3–11*

load/boot: *Vol. 3 3–15*

## kernel memory

maximum used by RFS users: *Vol. 3 4–117*

minimum available parameter: *Vol. 3 4–55*

keyboard mapping, change: *Vol. 1 D-4*

keyword: *Vol. 2 5-39*

---

**L**

L5WatchDogPeriod, parameters: *Vol. 3 4-40*

L5WatchDogPFRPeriod, parameters: *Vol. 3 4-39*

level of logging, sendmail: *Vol. 1 F-25*

limit

display for print queue priority: *Vol. 2 5-3*

for parameters: *Vol. 3 1-6*

set for priority: *Vol. 2 5-3*

limit load, sendmail: *Vol. 1 F-24*

Limits file: *Vol. 2 9-53*

line characteristics, change to 8 bit: *Vol. 1 D-2*

link, symbolic: *Vol. 1 A-5*

list

character set: *Vol. 2 5-26*

file system information: *Vol. 1 4-15*

installed file system types: *Vol. 1 4-6*

print wheel: *Vol. 2 5-25*

software packages: *Vol. 1 1-30*

load

kernel: *Vol. 3 3-15*

manage for printer: *Vol. 2 4-41*

log, print requests: *Vol. 2 5-46*

log file, check size of: *Vol. 2 9-68, 9-70*

log files, combine for BNU: *Vol. 2 9-68, 9-69*

log in: *Vol. 1 1-2*

log level, sendmail: *Vol. 1 F-25*

logs, for BNU: *Vol. 2 8-16*

LOTSFREE: *Vol. 3 3-32, 4-42, 6-5*

lp package: *Vol. 2 4-11*

## LP Print Service

customize: *Vol. 2 6–12*  
 description: *Vol. 2 4–2*  
 display filter: *Vol. 2 5–44*  
 filter: *Vol. 2 5–32*  
 install: *Vol. 2 4–11*  
 remove filter: *Vol. 2 5–44*  
 start: *Vol. 2 4–39*  
 stop: *Vol. 2 4–39*

lpfilter: *Vol. 2 5–44*

lpforms: *Vol. 2 5–15, 5–16*

lpstat: *Vol. 2 4–37*

---

**M**

machine architecture, parameter: *Vol. 3 4–5*

macros, sendmail: *Vol. 1 E–17*

mail queue, sendmail: *Vol. 1 F–11*

mailer declaration, sendmail: *Vol. 1 E–17*

mailer flag, sendmail: *Vol. 1 F–55*

## maintain

postScript: *Vol. 2 6–7*

PostScript fonts: *Vol. 2 6–8*

PostScript printers: *Vol. 2 6–6*

maintenance diskette, copy: *Vol. 1 2–6*

Maintenance File System, access: *Vol. 1 2–11*

maintenance file system: *Vol. 1 2–2*

check file system: *Vol. 1 3–40*

maintenance flex diskette, copy: *Vol. 1 2–7*

major device numbers: *Vol. 1 B–2*

make, file system: *Vol. 1 4–7*

**manage**

printer faults: *Vol. 2 5–7*  
printer load: *Vol. 2 4–41*  
queue priorities: *Vol. 2 5–2*  
quotas: *Vol. 1 4–21*

**map**

change keyboard: *Vol. 1 D–4*  
change screen: *Vol. 1 D–6*  
character set names: *Vol. 2 5–26*  
map table, for PostScript fonts: *Vol. 2 6–10*  
mapkey command: *Vol. 1 D–4*  
mapping, Name-to-Address: *Vol. 2 7–10*  
mapscrm command: *Vol. 1 D–6*  
mark, bad blocks: *Vol. 2 1–19*  
MAXAIOS: *Vol. 3 3–40, 4–43*  
MAXCLSYSPRI: *Vol. 3 3–42, 4–44*  
MAXDMAPAGE: *Vol. 3 3–38, 4–45*  
MAXFC: *Vol. 3 3–32*  
MAXGDP: *Vol. 3 3–37, 4–46*  
maximum click number for DMA, parameter: *Vol. 3 4–45*  
MAXMINOR: *Vol. 3 3–29, 4–47*  
MAXPMEM: *Vol. 3 3–29, 4–48*  
MAXSC: *Vol. 3 3–32*  
MAXSEPGCNT: *Vol. 3 3–33*  
MAXSERVE: *Vol. 3 3–37, 4–49*  
MAXSLICE: *Vol. 3 3–29, 4–50*  
MAXUMEM: *Vol. 3 3–32*  
MAXUP: *Vol. 3 3–29, 4–51*  
MAXWCONSMMSG: *Vol. 3 3–34, 4–52*  
MAXWERRMSG: *Vol. 3 3–34, 4–53*  
MAXWTRCMSG: *Vol. 3 3–34, 4–54*

## memory

- management: *Vol. 3 6–2*
- maximum pages lock in: *Vol. 3 4–105*
- maximum physical to use: *Vol. 3 4–48*
- minimum available for kernel: *Vol. 3 4–55*
- minimum free parameter: *Vol. 3 4–58*
- parameter for amount of sufficient free: *Vol. 3 4–42*
- parameter for desired free: *Vol. 3 4–9*
- save to dump area: *Vol. 1 3–35*

memory dump, customize: *Vol. 1 1–44*

menu, delete: *Vol. 1 C–31*

## menu entry

- change: *Vol. 1 C–18*
- create: *Vol. 1 C–18*

## menus

- create an entry: *Vol. 1 C–18*
- delete task entry: *Vol. 1 C–31*

message, maximum size parameter: *Vol. 3 4–62*

message collection, sendmail: *Vol. 1 E–15*

message delivery, sendmail: *Vol. 1 E–16*

## message queue

- maximum length parameter: *Vol. 3 4–63*
- maximum parameter: *Vol. 3 4–64*

## message segment

- maximum parameter: *Vol. 3 4–65*
- size parameter: *Vol. 3 4–66*

MEVDIRENTS: *Vol. 3 3–41*

MEVEXITS: *Vol. 3 3–41*

MEVEXPRS: *Vol. 3 3–41*

MEVEXREFS: *Vol. 3 3–41*

MEVKEYS: *Vol. 3 3–41*

MEVQUEUES: *Vol. 3 3–41*

MEVRETRYs: *Vol. 3 3–41*  
MEVSEXPRs: *Vol. 3 3–41*  
MEVSIGs: *Vol. 3 3–41*  
MEVSTERMs: *Vol. 3 3–41*  
MEVSTRDs: *Vol. 3 3–41*  
MEVTERMs: *Vol. 3 3–41*  
MEVTIDs: *Vol. 3 3–41*  
MINAIOS: *Vol. 3 3–40*  
MINAKMEM: *Vol. 3 3–32, 4–55, 6–8*  
MINARMEM: *Vol. 3 3–32, 4–56, 6–8*  
MINASMEM: *Vol. 3 3–32, 4–57, 6–8*  
MINFREE: *Vol. 3 3–32, 4–58, 6–6*  
MINHIDUSTK: *Vol. 3 3–32*  
minor device, number configured for log driver: *Vol. 3 4–86*  
minor device numbers: *Vol. 1 B–2*  
minor number, largest allowed on system: *Vol. 3 4–47*  
MINPAGEFREE: *Vol. 3 3–32, 4–59*  
MINSERVE: *Vol. 3 3–37, 4–60*  
MINUSTKGAP: *Vol. 3 3–32*  
mkfs command: *Vol. 1 4–9*  
MNR\_ON: *Vol. 3 3–39*  
mode, sendmail files: *Vol. 1 F–25*  
modem  
    configure on serial controller: *Vol. 2 3–25*  
    configure on serial port: *Vol. 2 3–15*  
Modems, configure: *Vol. 2 3–29*  
module, maximum pushed on stream: *Vol. 3 4–98*  
modules: *Vol. 3 3–17*  
monitor, disk usage: *Vol. 3 7–7*  
mount, file system: *Vol. 1 4–11*



move

print request to head of queue: *Vol. 2 5–6*

request in print queue: *Vol. 2 5–3*

MSGMAP: *Vol. 3 3–35, 4–61*

MSGMAX: *Vol. 3 3–35, 4–62*

MSGMNB: *Vol. 3 3–35, 4–63*

MSGMNI: *Vol. 3 3–35, 4–64*

MSGMSEG: *Vol. 3 4–65*

MSGSEG: *Vol. 3 3–35*

MSGSSZ: *Vol. 3 3–35, 4–66*

MSGTQL: *Vol. 3 3–35, 4–67*

multi-user state: *Vol. 1 1–5*

---

## N

NADVERTISE: *Vol. 3 3–37, 4–68*

NAIOPROC: *Vol. 3 3–40*

NAIOSYS: *Vol. 3 3–40, 4–69*

name

character set: *Vol. 2 5–25*

device: *Vol. 1 B–5*

print wheel: *Vol. 2 5–25*

Name-to-Address mapping: *Vol. 2 7–10*

national keyboard, map: *Vol. 1 D–4*

NAUTOPUSH: *Vol. 3 3–34, 4–70*

NAUTOUP: *Vol. 3 3–29, 4–71*

NBLK parameters: *Vol. 3 3–33*

NBUF: *Vol. 3 3–29, 4–72*

NCALL: *Vol. 3 3–29, 4–74*

NCDEXTENT, parameters: *Vol. 3 4–75*

NCDFILSYS, parameters: *Vol. 3 4–76*

NCDINODE, parameters: *Vol. 3 4–77*

NCLIST: *Vol. 3* 3–29, 4–78  
NCPU: *Vol. 3* 3–29, 4–79  
NCPYRIGHT: *Vol. 3* 3–39, 4–80  
NCR TOWERS, copy files to: *Vol. 1* 6–25  
NDQUOTE: *Vol. 3* 3–31, 4–81  
NEMAP: *Vol. 3* 3–38, 4–82  
NETPATH: *Vol. 2* 7–2  
network problems, printer: *Vol. 2* 4–55  
network selection: *Vol. 2* 7–2  
networking daemons: *Vol. 2* 8–8  
NFILE: *Vol. 3* 3–29  
NGROUPS\_MAX: *Vol. 3* 3–29, 4–83  
NHBUF: *Vol. 3* 3–29, 4–84  
NINODE: *Vol. 3* 3–31, 4–85  
NKDVTTY: *Vol. 3* 3–39  
NLOCAL: *Vol. 3* 3–37  
NLOG: *Vol. 3* 3–34, 4–86  
NMOUNT: *Vol. 3* 3–29, 4–87  
NMUXLINK: *Vol. 3* 3–33  
NODE: *Vol. 3* 3–46, 4–88  
node name  
    parameter: *Vol. 3* 4–88  
    set for running kernel: *Vol. 2* 9–9  
NOFILES: *Vol. 3* 3–29  
NOTPGOVERFLOW: *Vol. 3* 3–44, 4–89  
NPBUF: *Vol. 3* 3–29  
NPROC: *Vol. 3* 3–29, 4–90  
NQUEUE: *Vol. 3* 3–33  
NRCVD: *Vol. 3* 3–37, 4–91  
NRDUSER: *Vol. 3* 3–37, 4–92  
NREGION: *Vol. 3* 3–29

NREMOTE: *Vol. 3* 3–37  
 NRNODE: *Vol. 3* 3–31, 4–93  
 NSSINODE: *Vol. 3* 3–31  
 NSCRN: *Vol. 3* 3–38, 4–94  
 NSNDD: *Vol. 3* 3–37, 4–95  
 NSRMOUNT: *Vol. 3* 3–37, 4–96  
 NSTREAM: *Vol. 3* 3–33  
 NSTREVENT: *Vol. 3* 3–33  
 NSTRPHASH: *Vol. 3* 3–34, 4–97  
 NSTRPUSH: *Vol. 3* 3–33, 4–98  
 NUMSAD: *Vol. 3* 3–34, 4–99  
 NUMSP: *Vol. 3* 3–34, 4–100  
 NUMSXT: *Vol. 3* 3–39, 4–101  
 NUMTIM: *Vol. 3* 3–34, 4–102  
 NUMTRW: *Vol. 3* 3–34, 4–103  
 NUMXT: *Vol. 3* 3–39, 4–104



offline: *Vol. 1* 1–34  
 online: *Vol. 1* 1–34  
 option  
     for print service: *Vol. 2* 4–27  
     set for sendmail: *Vol. 1* E–18  
 organization, file system: *Vol. 1* A–2, A–6  
 osa login: *Vol. 1* 1–2  
 output illegible, printers: *Vol. 2* 4–50

---

**P**

packages, software: *Vol. 1* 1–30

page

maximum locked in memory parameter: *Vol. 3* 4–105

minimum free parameter: *Vol. 3* 4–59

PAGES\_UNLOCK: *Vol. 3* 3–32, 4–105

parameter

AFFIN\_ON: *Vol. 3* 3–42, 4–2

AFFINDECAY: *Vol. 3* 3–42, 4–3

affinity scheduling: *Vol. 3* 3–42

AGEINTERVAL: *Vol. 3* 3–32

AIOTIMEOUT: *Vol. 3* 3–40, 4–4

ARCHITECTURE: *Vol. 3* 3–47, 4–5

ARG\_MAX: *Vol. 3* 3–29, 4–6

BDFLUSHR: *Vol. 3* 3–29

BMAX: *Vol. 3* 3–44

BUFHWM: *Vol. 3* 3–29, 4–7

CMAX: *Vol. 3* 3–44

CMF: *Vol. 3* 3–39, 4–8

COM2CONS: *Vol. 3* 3–39

configuration: *Vol. 3* 3–25

D0387CR3: *Vol. 3* 3–38

DESFREE: *Vol. 3* 3–32, 4–9, 6–5

DMAABLEBUF: *Vol. 3* 3–38, 4–10

DMAEXCL: *Vol. 3* 3–38, 4–11

DO386B1: *Vol. 3* 3–38

DSTFLAG: *Vol. 3* 3–38, 4–12

EVDATA: *Vol. 3* 3–41

EVFNHTS: *Vol. 3* 3–41

EVMAXDPE: *Vol. 3* 3–41

EVMAXETERMS: *Vol. 3* 3–41

EVMAXEV: *Vol. 3* 3–41

EVMAXMEM: *Vol. 3* 3–41

EVMAXTRAPS: *Vol. 3* 3–41

EVTIDHTS: *Vol. 3 3–41*  
FDFLUSHR: *Vol. 3 3–29, 4–13*  
FLCKREC: *Vol. 3 3–29, 4–14*  
GPGSHI: *Vol. 3 3–32*  
GPGSLO: *Vol. 3 3–32, 4–15, 6–7*  
GPGSMSK: *Vol. 3 3–32*  
HCORLIM: *Vol. 3 3–43, 4–16*  
HCPULIM: *Vol. 3 3–43, 4–17*  
HDATLIM: *Vol. 3 3–43, 4–18*  
HFNOLIM: *Vol. 3 3–43, 4–19*  
HFSZLIM: *Vol. 3 3–43, 4–20*  
HRTIME: *Vol. 3 3–42, 4–21*  
HRVTIME: *Vol. 3 3–42, 4–22*  
HSTKLIM: *Vol. 3 3–43, 4–23*  
HVMMLIM: *Vol. 3 3–43, 4–24*  
HW\_PROVIDER: *Vol. 3 3–47, 4–25*  
HW\_SERIAL: *Vol. 3 3–47, 4–26*  
ILDMAXOPENS: *Vol. 3 3–45*  
KDBSYMSIZE: *Vol. 3 3–38, 4–31*  
L5WatchDogPeriod: *Vol. 3 4–40*  
L5WatchDogPFRPeriod: *Vol. 3 4–39*  
LOTSFREE: *Vol. 3 3–32, 4–42, 6–5*  
MAXAIOS: *Vol. 3 3–40, 4–43*  
MAXCLSYSPRI: *Vol. 3 3–42, 4–44*  
MAXDMAPAGE: *Vol. 3 3–38, 4–45*  
MAXFC: *Vol. 3 3–32*  
MAXGDP: *Vol. 3 3–37, 4–46*  
MAXMINOR: *Vol. 3 3–29, 4–47*  
MAXPMEM: *Vol. 3 3–29, 4–48*  
MAXSC: *Vol. 3 3–32*  
MAXSEPGCNT: *Vol. 3 3–33*  
MAXSERVE: *Vol. 3 3–37, 4–49*  
MAXSLICE: *Vol. 3 3–29, 4–50*  
MAXUMEM: *Vol. 3 3–32*  
MAXUP: *Vol. 3 3–29, 4–51*

MAXWCONSMMSG: *Vol. 3 3–34, 4–52*  
MAXWERRMSG: *Vol. 3 3–34, 4–53*  
MAXWTRCMSG: *Vol. 3 3–34, 4–54*  
MEVDIRENTS: *Vol. 3 3–41*  
MEVEXITS: *Vol. 3 3–41*  
MEVEXPRS: *Vol. 3 3–41*  
MEVEXREFS: *Vol. 3 3–41*  
MEVKEVS: *Vol. 3 3–41*  
MEVQUEUEES: *Vol. 3 3–41*  
MEVRETRYs: *Vol. 3 3–41*  
MEVSEXPRS: *Vol. 3 3–41*  
MEVSIGS: *Vol. 3 3–41*  
MEVSTERMS: *Vol. 3 3–41*  
MEVSTRDS: *Vol. 3 3–41*  
MEVTERMS: *Vol. 3 3–41*  
MEVTIDS: *Vol. 3 3–41*  
MINAIOS: *Vol. 3 3–40*  
MINAKMEM: *Vol. 3 3–32, 4–55, 6–8*  
MINARMEM: *Vol. 3 3–32, 4–56, 6–8*  
MINASMEM: *Vol. 3 3–32, 4–57, 6–8*  
MINFREE: *Vol. 3 3–32, 4–58, 6–6*  
MINHIDUSTK: *Vol. 3 3–32*  
MINPAGEFREE: *Vol. 3 3–32, 4–59*  
MINSERVE: *Vol. 3 3–37, 4–60*  
MINUSTKGAP: *Vol. 3 3–32*  
MNR\_ON: *Vol. 3 3–39*  
MSGMAP: *Vol. 3 3–35, 4–61*  
MSGMAX: *Vol. 3 3–35, 4–62*  
MSGMNB: *Vol. 3 3–35, 4–63*  
MSGMNI: *Vol. 3 3–35, 4–64*  
MSGSEG: *Vol. 3 3–35, 4–65*  
MSGSSZ: *Vol. 3 3–35, 4–66*  
MSGTQL: *Vol. 3 3–35, 4–67*  
NADVERTISE: *Vol. 3 3–37, 4–68*  
NAIOPROC: *Vol. 3 3–40*

NAIOSYS: *Vol. 3* 3–40, 4–69  
NAUTOPUSH: *Vol. 3* 3–34, 4–70  
NAUTOUP: *Vol. 3* 3–29, 4–71  
NBLK: *Vol. 3* 3–33  
NBUF: *Vol. 3* 3–29, 4–72  
NCALL: *Vol. 3* 3–29, 4–74  
NCDEXTENT: *Vol. 3* 4–75  
NCDFILSYS: *Vol. 3* 4–76  
NCDINODE: *Vol. 3* 4–77  
NCLIST: *Vol. 3* 3–29, 4–78  
NCPU: *Vol. 3* 3–29, 4–79  
NCPYRIGHT: *Vol. 3* 3–39, 4–80  
NDQUOT: *Vol. 3* 3–31, 4–81  
NEMAP: *Vol. 3* 3–38, 4–82  
NFILE: *Vol. 3* 3–29  
NGROUPS\_MAX: *Vol. 3* 3–29, 4–83  
NHBUF: *Vol. 3* 3–29, 4–84  
NINODE: *Vol. 3* 3–31, 4–85  
NKDVTTY: *Vol. 3* 3–39  
NLOCAL: *Vol. 3* 3–37  
NLOG: *Vol. 3* 3–34, 4–86  
NMOUNT: *Vol. 3* 3–29, 4–87  
NMUXLINK: *Vol. 3* 3–33  
NODE: *Vol. 3* 3–46, 4–88  
NOFILES: *Vol. 3* 3–29  
NOTPGOVERFLOW: *Vol. 3* 3–44, 4–89  
NPBUF: *Vol. 3* 3–29  
NPROC: *Vol. 3* 3–29, 4–90  
NQUEUE: *Vol. 3* 3–33  
NRCVD: *Vol. 3* 3–37, 4–91  
NRDUSER: *Vol. 3* 3–37, 4–92  
NREGION: *Vol. 3* 3–29  
NREMOTE: *Vol. 3* 3–37  
NRNODE: *Vol. 3* 3–31, 4–93  
NSSINODE: *Vol. 3* 3–31

NSCRN: *Vol. 3* 3–38, 4–94  
NSNDD: *Vol. 3* 3–37, 4–95  
NSRMOUNT: *Vol. 3* 3–37, 4–96  
NSTREAM: *Vol. 3* 3–33  
NSTREVENT: *Vol. 3* 3–33  
NSTRPHASH: *Vol. 3* 3–34, 4–97  
NSTRPUSH: *Vol. 3* 3–33, 4–98  
NUMSAD: *Vol. 3* 3–34, 4–99  
NUMSP: *Vol. 3* 3–34, 4–100  
NUMSXT: *Vol. 3* 3–39, 4–101  
NUMTIM: *Vol. 3* 3–34, 4–102  
NUMTRW: *Vol. 3* 3–34, 4–103  
NUMXT: *Vol. 3* 3–39, 4–104  
PAGES\_UNLOCK: *Vol. 3* 3–32, 4–105  
PGOVERFLOW: *Vol. 3* 3–44, 4–106  
PIOMAP: *Vol. 3* 3–30, 4–107  
PIOMAXSZ: *Vol. 3* 3–30, 4–108  
PIOSEGSZ: *Vol. 3* 3–38, 4–109  
PRFMAX: *Vol. 3* 3–39, 4–110  
PUTBUFSZ: *Vol. 3* 3–30, 4–111  
RCACHETIME: *Vol. 3* 3–37  
RCHACHETIME: *Vol. 3* 4–114  
RCMF: *Vol. 3* 3–39, 4–115  
REL: *Vol. 3* 3–46, 4–116  
RF\_MAXKMEM: *Vol. 3* 3–37, 4–117  
RFHEAP: *Vol. 3* 3–37  
RFS\_VHIGH: *Vol. 3* 3–37  
RFS\_VLOW: *Vol. 3* 3–37  
RIDEOUT: *Vol. 3* 3–39  
ROOTFSTYPE: *Vol. 3* 4–118  
RSTCHOWN: *Vol. 3* 3–30, 4–119  
RTMAXPRI: *Vol. 3* 3–42  
RTNPROCS: *Vol. 3* 3–42, 4–120  
S52KNBUF: *Vol. 3* 3–31  
S52KNHBUF: *Vol. 3* 3–31



SANECNT: *Vol. 3* 3–39  
SANITYCLK: *Vol. 3* 3–38  
SATENABLED: *Vol. 3* 3–45, 4–121  
scheduler: *Vol. 3* 3–42  
SCORLIM: *Vol. 3* 3–43, 4–122  
SCPULIM: *Vol. 3* 3–43, 4–123  
SDATLIM: *Vol. 3* 3–43, 4–124  
SEGMAPSZ: *Vol. 3* 3–38, 4–125  
SEMAEM: *Vol. 3* 3–35, 4–126  
SEMMAP: *Vol. 3* 3–35, 4–127  
SEMMNI: *Vol. 3* 3–35, 4–128  
SEMMNS: *Vol. 3* 3–35, 4–129  
SEMMNU: *Vol. 3* 3–35, 4–130  
SEMMSL: *Vol. 3* 3–35, 4–131  
SEMOPM: *Vol. 3* 3–35, 4–132  
SEMUME: *Vol. 3* 3–35, 4–133  
SEVMVMX: *Vol. 3* 3–35, 4–134  
SFNOLIM: *Vol. 3* 3–43, 4–135  
SFSZLIM: *Vol. 3* 3–43, 4–136  
SHLBMAX: *Vol. 3* 3–30, 4–137  
SHMMAX: *Vol. 3* 3–36, 4–139  
SHMMIN: *Vol. 3* 3–36, 4–140  
SHMMNI: *Vol. 3* 3–36, 4–141  
SHM\_NAILED\_GID[1–9]: *Vol. 3* 4–138  
SHMSEG: *Vol. 3* 3–36, 4–142  
SOCK\_HIWATER: *Vol. 3* 3–34, 4–143  
SOCK\_LOWATER: *Vol. 3* 3–34, 4–144  
SPTMAP: *Vol. 3* 3–30  
SRPC\_DOMAIN: *Vol. 3* 3–46, 4–145  
SSTKLIM: *Vol. 3* 3–43, 4–146  
STRCTLSZ: *Vol. 3* 3–33, 4–147  
STRLOFRAC: *Vol. 3* 3–33  
STRMEDFRAC: *Vol. 3* 3–33  
STRMSGSZ: *Vol. 3* 3–33, 4–148  
STRTHRESH: *Vol. 3* 3–33, 4–149

SVMMLIM: *Vol. 3* 3-43, 4-150  
SYS: *Vol. 3* 3-46, 4-151  
TIM\_HIWATER: *Vol. 3* 3-34, 4-152  
TIM\_LOWATER: *Vol. 3* 3-34, 4-153  
timer: *Vol. 3* 3-42  
TIMEZONE: *Vol. 3* 3-38, 4-154  
TRACESZ: *Vol. 3* 3-30, 4-155  
TRW\_HIWATER: *Vol. 3* 3-34, 4-156  
TRW\_LOWATER: *Vol. 3* 3-34, 4-157  
TSMAXUPRI: *Vol. 3* 3-42, 4-158  
TSNPROCS: *Vol. 3* 3-42, 4-159  
UFSINODE: *Vol. 3* 3-31  
UFSNINODE: *Vol. 3* 4-161  
ULIMIT: *Vol. 3* 3-30, 4-162  
USANEON: *Vol. 3* 3-39  
VER: *Vol. 3* 3-46, 4-163  
VHNDFRAC: *Vol. 3* 3-32  
XSDSEGS: *Vol. 3* 3-38, 4-164  
XSDSLOTS: *Vol. 3* 3-38, 4-165  
XSEMMAX: *Vol. 3* 3-38, 4-166  
parity setting, printer: *Vol. 2* 4-51  
partition  
    change active: *Vol. 2* 1-24  
    disk: *Vol. 2* 1-8  
password, recover for root: *Vol. 1* 3-10  
password file: *Vol. 1* 1-28  
password shadowing: *Vol. 1* 1-28  
PATH variables: *Vol. 3* 8-3  
pbind: *Vol. 1* 1-35  
PBS, power backup system: *Vol. 1* 1-37  
Permissions file: *Vol. 2* 9-35  
    examples: *Vol. 2* 9-45  
PGOVERFLOW: *Vol. 3* 3-44, 4-106

phases of fsck, on ufs file systems: *Vol. 1* 5–47  
 physical connection, for BNU: *Vol. 2* 9–7  
 physical memory, maximum amount to use: *Vol. 3* 4–48  
 pinfo: *Vol. 1* 1–33  
 PIOMAP: *Vol. 3* 3–30, 4–107  
 PIOMAXSZ: *Vol. 3* 3–30, 4–108  
 PIOSEGSZ: *Vol. 3* 3–38, 4–109  
 pkgadd: *Vol. 2* 4–11  
 pmadm(1M): *Vol. 2* 3–7  
 port monitor  
     administrative file: *Vol. 2* 3–5  
     itymon: *Vol. 2* 3–8  
 PostScript, maintain: *Vol. 2* 6–6  
 PostScript fonts  
     host-resident: *Vol. 2* 6–9  
     map table: *Vol. 2* 6–10  
     printer-resident: *Vol. 2* 6–9  
 PostScript printer: *Vol. 2* 6–2  
     install: *Vol. 2* 6–6, 6–7  
 power backup systems, PBS: *Vol. 1* 1–37  
 powerfail, strategy: *Vol. 1* 1–37  
 PRFMAX: *Vol. 3* 3–39, 4–110  
 print, sendmail mail queue: *Vol. 1* F–11  
 print request  
     hold: *Vol. 2* 5–5  
     move in queue: *Vol. 2* 5–3  
     move to head of queue: *Vol. 2* 5–6  
 print service  
     customize: *Vol. 2* 6–12  
     install: *Vol. 2* 4–11  
     start: *Vol. 2* 4–39  
     stop: *Vol. 2* 4–39

- print service options: *Vol. 2 4–27*
- print wheel: *Vol. 2 5–24*
  - list: *Vol. 2 5–25*
  - name: *Vol. 2 5–25*
- printer
  - adjust port characteristics: *Vol. 2 6–12*
  - character set: *Vol. 2 5–24*
  - configure: *Vol. 2 4–12*
  - default priority: *Vol. 2 5–3*
  - dial out failures: *Vol. 2 4–54*
  - display filter: *Vol. 2 5–44*
  - display status of: *Vol. 2 4–37*
  - faults: *Vol. 2 5–7*
  - filter: *Vol. 2 5–32*
  - filter defaults: *Vol. 2 5–45*
  - form: *Vol. 2 5–12*
  - idle: *Vol. 2 4–55*
  - interface program: *Vol. 2 6–18*
  - managing load: *Vol. 2 4–41*
  - networking problems: *Vol. 2 4–55*
  - print wheel: *Vol. 2 5–24*
  - queue priorities: *Vol. 2 5–2*
  - remove filter: *Vol. 2 5–44*
  - troubleshooting: *Vol. 2 4–49*
- printer–resident fonts: *Vol. 2 6–9*
- priorities, print queue: *Vol. 2 5–2*
- priority
  - default: *Vol. 2 5–3*
  - display limits and defaults: *Vol. 2 5–3*
  - sendmail queue: *Vol. 1 F–23*
  - set limits: *Vol. 2 5–3*

## process

analyze with time: *Vol. 3 1–45*

analyze with timex: *Vol. 3 1–47*

bind: *Vol. 1 1–35*

maximum address space parameter: *Vol. 3 4–150*

maximum created parameter: *Vol. 3 4–90*

maximum group parameter: *Vol. 3 4–83*

maximum size of stack segment: *Vol. 3 4–146*

size of data segment parameter: *Vol. 3 4–124*

process accounting: *Vol. 3 1–35*

shutdown: *Vol. 3 1–36*

process binding, display information: *Vol. 1 1–35*

process table, number of entries to allocate: *Vol. 3 4–90*

## processor

activate: *Vol. 1 1–33*

deactivate: *Vol. 1 1–33*

processors, display status: *Vol. 1 1–33*

profile system: *Vol. 3 1–49*

profiler command: *Vol. 3 1–49*

## protocol

file transfer: *Vol. 2 9–60*

for devices: *Vol. 2 9–27*

prvtoc command: *Vol. 2 1–23*

public area, clean up: *Vol. 2 9–68, 9–69*

PUTBUFSZ: *Vol. 3 3–30, 4–111*

---

**Q**

## queue

move request: *Vol. 2 5–3*

sendmail: *Vol. 1 F–11*

queue interval, sendmail: *Vol. 1 F–19*

queue priorities: *Vol. 2 5–2*  
    sendmail: *Vol. 1 F–23*  
queued message, sendmail: *Vol. 1 E–16*  
quot command: *Vol. 1 4–22*  
quota structure, parameter for ufs: *Vol. 3 4–81*  
quotacheck command: *Vol. 1 4–21*  
quotaoff command: *Vol. 1 4–22*  
quotaon command: *Vol. 1 4–22*  
quotas: *Vol. 1 4–20*  
    manage: *Vol. 1 4–21*  
    set: *Vol. 1 4–20*  
    turn off: *Vol. 1 4–22*  
    turn on: *Vol. 1 4–22*

---

**R**

raw devices  
    backup: *Vol. 1 6–14*  
    restore: *Vol. 1 7–8*  
raw I/O, size of segment parameter: *Vol. 3 4–109*  
RCACHETIME: *Vol. 3 3–37, 4–114*  
RCMF: *Vol. 3 3–39, 4–115*  
real-time processes, parameter: *Vol. 3 4–120*  
receive, printer fault alert: *Vol. 2 5–8*  
receive descriptor, maximum user entires parameter: *Vol. 3 4–92*  
receive descriptors, number configured parameter: *Vol. 3 4–91*  
record locking regions, parameter for maximum number: *Vol. 3 4–14*

**recover**

*/etc/passwd: Vol. 1 3–13*

*/etc/shadow: Vol. 1 3–17*

*BOOT program: Vol. 1 3–29*

*files from installation tape: Vol. 1 3–21*

*files in the /stand file system: Vol. 1 3–2*

*root password: Vol. 1 3–10*

*sector 0: Vol. 1 3–25*

*unix: Vol. 1 3–6*

**recover diskette, copy:** *Vol. 1 2–6*

**reference diskette:** *Vol. 1 2–2*

**REL:** *Vol. 3 3–46, 4–116*

**release, parameter:** *Vol. 3 4–116*

**remote installation**

*overview: Vol. 1 1–47*

*setting up remote console: Vol. 1 1–49*

*setting up the equipment: Vol. 1 1–48*

**remote mounts, maximum parameter:** *Vol. 3 4–96*

**remote,unknown file:** *Vol. 2 9–57*

**remove**

*filter: Vol. 2 5–44*

*inactive file: Vol. 3 7–11*

*software packages: Vol. 1 1–31*

**repair, file system:** *Vol. 1 5–4*

**replace, failed disk:** *Vol. 2 1–35*

**repquota command:** *Vol. 1 4–22*

**request, hold:** *Vol. 2 5–5*

**request log:** *Vol. 2 5–46*

**restore**

*corrupted root file system: Vol. 1 7–21*

*entire disk: Vol. 1 7–10*

*file systems: Vol. 1 7–2*

*raw devices: Vol. 1 7–8*

restoring, entire system: *Vol. 1* 7–20  
 restricted file ownership flag: *Vol. 3* 4–119  
 retention a cartridge tape: *Vol. 1* 6–26  
 RF\_MAXKMEM: *Vol. 3* 3–37, 4–117  
 RFFHEAP: *Vol. 3* 3–37  
 RFS\_VHIGH: *Vol. 3* 3–37  
 RFS\_VLOW: *Vol. 3* 3–37  
 RIDEOUT: *Vol. 3* 3–39  
 root, recover password: *Vol. 1* 3–10  
 root file system  
     parameter: *Vol. 3* 4–118  
     restore corrupted: *Vol. 1* 7–21  
 root login: *Vol. 1* 1–2  
 root logins, from terminals: *Vol. 1* 1–3  
 ROOTFSTYPE: *Vol. 3* 4–118  
 RSTCHOWN: *Vol. 3* 3–30, 4–119  
 RTMAXPRI: *Vol. 3* 3–42  
 RTNPROCS: *Vol. 3* 3–42, 4–120  
 run level, change: *Vol. 1* 1–5  
 runacct command: *Vol. 3* 1–35

---

**S**

s5 file system type: *Vol. 1* A–10  
 S52KNBUF: *Vol. 3* 3–31  
 S52KNHBUF: *Vol. 3* 3–31  
 SAC. *See* service access controller  
 sacadm(1M): *Vol. 2* 3–5  
 sadc: *Vol. 3* 1–5  
     output: *Vol. 3* 1–6  
 SAF. *See* service access facility  
 SANECNT: *Vol. 3* 3–39



SANITYCLK: *Vol. 3* 3–38  
 sar, analyze activity data: *Vol. 3* 1–7  
 SATENABLED: *Vol. 3* 3–45, 4–121  
 save dump area to media: *Vol. 1* 3–37  
 save memory to dump area: *Vol. 1* 3–35  
 scheduler parameters: *Vol. 3* 3–42  
 SCORLIM: *Vol. 3* 3–43, 4–122  
 SCPULIM: *Vol. 3* 3–43, 4–123  
 screen mapping, change: *Vol. 1* D–6  
 SCSI controller, add: *Vol. 2* 2–27, 2–34  
 SDATLIM: *Vol. 3* 3–43, 4–124  
 sector 0, recover: *Vol. 1* 3–25  
 Secure RPC Domain, parameter: *Vol. 3* 4–145  
 security audit trail, parameter: *Vol. 3* 4–121  
 SEGMAPSZ: *Vol. 3* 3–38, 4–125  
 SEMAEM: *Vol. 3* 3–35, 4–126  
 semaphore  
     adjust-on-exit value: *Vol. 3* 4–126  
     maximum in system: *Vol. 3* 4–129  
     maximum operations per semop: *Vol. 3* 4–132  
     maximum per semaphore identifier: *Vol. 3* 4–131  
     maximum value parameter: *Vol. 3* 4–134  
     number entries in control map: *Vol. 3* 4–127  
 SEMMAP: *Vol. 3* 3–35, 4–127  
 SEMMNI: *Vol. 3* 3–35, 4–128  
 SEMMNS: *Vol. 3* 3–35, 4–129  
 SEMMNU: *Vol. 3* 3–35, 4–130  
 SEMMSL: *Vol. 3* 3–35, 4–131  
 SEMOPM: *Vol. 3* 3–35, 4–132  
 semaphore, maximum identifiers in kernel: *Vol. 3* 4–128  
 SEMUME: *Vol. 3* 3–35, 4–133

SEMMVMX: *Vol. 3* 3–35, 4–134

send descriptors, maximum parameter: *Vol. 3* 4–95

sendmail

alias database: *Vol. 1* F–15

aliasing: *Vol. 1* E–14

alternate configuration file: *Vol. 1* F–20

arguments: *Vol. 1* E–13

argurments: *Vol. 1* F–19

building a configuration file: *Vol. 1* F–39

command line flags: *Vol. 1* F–48

configuration: *Vol. 1* E–16

configuration file: *Vol. 1* F–10, F–27

configuration options: *Vol. 1* F–50

daemon mode: *Vol. 1* F–19

debugging: *Vol. 1* F–20

delivery mode: *Vol. 1* F–24

design goals: *Vol. 1* E–5

file mode: *Vol. 1* F–25

forcing the queue: *Vol. 1* F–19

forwarding: *Vol. 1* E–14

forwarding files: *Vol. 1* F–18

header declarations: *Vol. 1* E–17

inclusion: *Vol. 1* E–15

installing: *Vol. 1* F–4

interfaces: *Vol. 1* E–8

limiting load: *Vol. 1* F–24

logging level: *Vol. 1* F–25

macros: *Vol. 1* E–17

mail queue: *Vol. 1* F–11

mailer declarations: *Vol. 1* E–17

mailer flags: *Vol. 1* F–55

message collection: *Vol. 1* E–15

message delivery: *Vol. 1* E–16

queue interval: *Vol. 1* F–19

- queue priorities: *Vol. 1 F-23*
- queued messages: *Vol. 1 E-16*
- set options: *Vol. 1 E-18*
- system log: *Vol. 1 F-10*
- tuning: *Vol. 1 F-21*
- serial controller
  - configure modem on: *Vol. 2 3-25*
  - configure terminal on: *Vol. 2 3-22*
  - install: *Vol. 2 2-18*
- serial number, parameter for: *Vol. 3 4-26*
- serial port
  - configure a second: *Vol. 2 3-20*
  - configure modem on: *Vol. 2 3-15*
  - configure terminal on: *Vol. 2 3-13*
- server
  - maximum handling remote requests: *Vol. 3 4-49*
  - minimum running on system: *Vol. 3 4-60*
- service access controller: *Vol. 2 3-3*
  - administrative file: *Vol. 2 3-4*
- service access facility: *Vol. 2 3-2*
- set
  - node name for running kernel: *Vol. 2 9-9*
  - priority limits: *Vol. 2 5-3*
  - quotas: *Vol. 1 4-20*
- set up
  - link for BNU: *Vol. 2 9-5*
  - virtual terminals: *Vol. 1 1-3*
- setuname command: *Vol. 2 9-10*
- setup
  - equipment for remote installation: *Vol. 1 1-48*
  - remote console: *Vol. 1 1-49*
- SFNOLIM: *Vol. 3 3-43, 4-135*
- SFSZLIM: *Vol. 3 3-43, 4-136*

- shadow file: *Vol. 1* 1–28
  - update: *Vol. 1* 1–29
- shadowing, password: *Vol. 1* 1–28
- shared data segment
  - attachments allowed: *Vol. 3* 4–165
  - number in system: *Vol. 3* 4–164
- shared library, maximum attached to process: *Vol. 3* 4–137
- shared memory
  - maximum attached segments parameter: *Vol. 3* 4–142
  - maximum identifiers parameter: *Vol. 3* 4–141
  - maximum segment size parameter: *Vol. 3* 4–139
  - minimum segment size parameter: *Vol. 3* 4–140
- shell layer, sessions in progress parameter: *Vol. 3* 4–101
- SHLBMAX: *Vol. 3* 3–30, 4–137
- SHM\_NAILED\_GID, parameters: *Vol. 3* 4–138
- SHMMAX: *Vol. 3* 3–36, 4–139
- SHMMIN: *Vol. 3* 3–36, 4–140
- SHMMNI: *Vol. 3* 3–36, 4–141
- SHMSEG: *Vol. 3* 3–36, 4–142
- shutdown: *Vol. 1* 1–5
- shutdown system: *Vol. 1* 1–7
- single user state: *Vol. 1* 1–5
- size, check for BNU log files: *Vol. 2* 9–68, 9–70
- slice
  - disk: *Vol. 2* 1–14
  - display information for hard disk: *Vol. 2* 1–23
- SOCK\_HIWATER: *Vol. 3* 4–143
- SOCK\_LOWATER: *Vol. 3* 4–144
- sockmod module
  - high water mark: *Vol. 3* 4–143
  - low water mark: *Vol. 3* 4–144
- sockmod module parameters: *Vol. 3* 3–34

- soft limit parameter: *Vol. 3* 1–6
- software packages: *Vol. 1* 1–30
  - install from flex: *Vol. 1* 1–30
  - install from tape: *Vol. 1* 1–30
  - list: *Vol. 1* 1–30
  - remove: *Vol. 1* 1–31
- SPTMAP: *Vol. 3* 3–30
- SRPC\_DOMAIN: *Vol. 3* 3–46, 4–145
- SSTKLIM: *Vol. 3* 3–43, 4–146
- stack segment, maximum size parameter: *Vol. 3* 4–146
- /stand
  - compact files: *Vol. 1* 4–18
  - recover files in: *Vol. 1* 3–2
- start, LP Print Service: *Vol. 2* 4–39
- start-of-day count: *Vol. 3* 7–10
- startup system: *Vol. 1* 1–7
- status
  - display for printers: *Vol. 2* 4–37
  - display for processors: *Vol. 1* 1–33
- stop
  - LP print service: *Vol. 2* 4–39
  - printer fault alert: *Vol. 2* 5–9
- strategy, powerfail: *Vol. 1* 1–37
- STRCTLSZ: *Vol. 3* 3–33, 4–147
- STRLOFRAC: *Vol. 3* 3–33
- STRMEDFRAC: *Vol. 3* 3–33
- STRMSGSZ: *Vol. 3* 3–33, 4–148
- STRTHRESH: *Vol. 3* 3–33, 4–149
- stty command: *Vol. 2* 4–31
- su login: *Vol. 1* 1–2
- support package: *Vol. 1* 1–13
- SVMMLIM: *Vol. 3* 3–43, 4–150

swap command: *Vol. 2 1–27*  
swap file, add: *Vol. 2 1–27*  
swap sapce, configure: *Vol. 2 1–25*  
symbolic links: *Vol. 1 A–5*  
SYS: *Vol. 3 3–46, 4–151*  
sysadm, menu hierarchy: *Vol. 1 C–2*  
sysadm login: *Vol. 1 1–2*  
system activity  
    analyze with process accounting: *Vol. 3 1–35*  
    analyze with sar: *Vol. 3 1–7*  
system log, sendmail: *Vol. 1 F–10*  
system message headers, maximum parameter: *Vol. 3 4–67*  
system name, parameter: *Vol. 3 4–151*  
system profiling: *Vol. 3 1–49*  
system release, parameter: *Vol. 3 4–116*  
system shutdown: *Vol. 1 1–7*  
system startup: *Vol. 1 1–7*  
system state, change: *Vol. 1 1–5*  
Systems file: *Vol. 2 9–11*

---

**T**

task entry, delete: *Vol. 1 C–31*  
template: *Vol. 2 5–38*  
terminal  
    configure on serial controller: *Vol. 2 3–22*  
    configure on serial port: *Vol. 2 3–13*  
    maximum xt supported parameter: *Vol. 3 4–104*  
    virtual: *Vol. 1 1–3*  
terminal line settings: *Vol. 2 3–11*  
terminfo databse, adjust: *Vol. 2 6–14*  
three-wire cable, configure serial port: *Vol. 2 3–35*

TIM\_HIWATER: *Vol. 3 4–152*  
 TIM\_LOWATER: *Vol. 3 4–153*  
 time command: *Vol. 3 1–45*  
 time slice, maximum value parameter: *Vol. 3 4–50*  
 time-sharing process, user priority parameter: *Vol. 3 4–158*  
 timer parameters: *Vol. 3 3–42*  
 timesharing process, number permitted: *Vol. 3 4–159*  
 timex command: *Vol. 3 1–47*  
 TIMEZONE: *Vol. 3 3–38, 4–154*  
 timod, maximum modes parameter: *Vol. 3 4–102*  
 timod module  
     high water mark: *Vol. 3 4–152*  
     low water mark: *Vol. 3 4–153*  
 timod module parameters: *Vol. 3 3–34*  
 timod module paramters: *Vol. 3 3–34*  
 tirdwr module  
     high water mark: *Vol. 3 4–156*  
     low water mark: *Vol. 3 4–157*  
 tirdwr module parameters: *Vol. 3 3–34*  
 tirdwr module paramters: *Vol. 3 3–34*  
 TRACESZ: *Vol. 3 3–30, 4–155*  
 troubleshooting, printer: *Vol. 2 4–49*  
 troubleshooting BNU  
     administrative problems: *Vol. 2 9–72*  
     faulty ACUs and modems: *Vol. 2 9–72*  
     no login: *Vol. 2 9–71*  
     out of space: *Vol. 2 9–71*

- troubleshooting printers
  - character set or font wrong: *Vol. 2 4–53*
  - dialout failures: *Vol. 2 4–54*
  - idle printers: *Vol. 2 4–55*
  - illegible output: *Vol. 2 4–50*
  - network problems: *Vol. 2 4–55*
  - no output: *Vol. 2 4–49*
  - spacing: *Vol. 2 4–52*
- troubleshooting with cu: *Vol. 2 9–73*
- troubleshooting with uutry: *Vol. 2 9–74*
- TRW\_HIWATER: *Vol. 3 4–156*
- TRW\_LOWATER: *Vol. 3 4–157*
- TSMAXUPRI: *Vol. 3 3–42, 4–158*
- TSNPROCS: *Vol. 3 3–42, 4–159*
- ttydefs: *Vol. 2 3–11*
- ttymon: *Vol. 2 3–8*
  - log files: *Vol. 2 3–10*
  - terminal line settings: *Vol. 2 3–11*
- tuning
  - basic steps: *Vol. 3 5–3*
  - bottlenecks: *Vol. 3 5–10*
  - reasons for: *Vol. 3 5–2*
  - sendmail: *Vol. 1 F–21*
- turn off, quotas: *Vol. 1 4–22*
- turn on quotas: *Vol. 1 4–22*
- type, identify for file system: *Vol. 1 4–6*
- types, file system: *Vol. 1 A–8*

---

## U

- ufs file system
  - check: *Vol. 1 5–38*
  - phases of fsck: *Vol. 1 5–47*



- ufs file system type: *Vol. 1* A-17
- UFSINODE: *Vol. 3* 3-31
- UFSNINODE: *Vol. 3* 4-161
- ULIMIT: *Vol. 3* 3-30, 4-162
- umount command: *Vol. 1* 4-14
- undeliverable BNU jobs, clean up: *Vol. 2* 9-68
- undo structure
  - maximum entries parameter: *Vol. 3* 4-133
  - maximum parameter: *Vol. 3* 4-130
- unix, recover: *Vol. 1* 3-6
- unmount, file system: *Vol. 1* 4-13
- usage of disk, monitor and control: *Vol. 3* 7-7
- USANEON: *Vol. 3* 3-39
- user
  - access to forms: *Vol. 2* 5-16
  - large space: *Vol. 3* 7-12
- user diagnostics diskette: *Vol. 1* 2-6
- uucheck command: *Vol. 2* 9-77
- uucp logins, add: *Vol. 2* 9-20
- uudemon.admin: *Vol. 2* 9-65
- uudemon.cleanup: *Vol. 2* 9-65
- uudemon.hour: *Vol. 2* 9-66
- uudemon.poll: *Vol. 2* 9-66
- uulog command: *Vol. 2* 9-77
- uuname command: *Vol. 2* 9-77
- uustat command: *Vol. 2* 9-68
- uutry command: *Vol. 2* 9-74

---

**V**

VER: *Vol. 3 3–46, 4–163*  
verify, BNU link: *Vol. 2 9–63*  
version of OS release, parameter: *Vol. 3 4–163*  
vfstab file: *Vol. 1 4–4*  
VHNDFRAC: *Vol. 3 3–32*  
virtual circuits, number open on network: *Vol. 3 4–46*  
virtual terminal, number permitted parameter: *Vol. 3 4–94*  
virtual terminals: *Vol. 1 1–3*  
    accessing: *Vol. 1 1–3*  
    closing: *Vol. 1 1–4*  
    setting up: *Vol. 1 1–3*

---

**W**

write  
    dialing information: *Vol. 2 9–30*  
    help messages: *Vol. 1 C–8*  
    interface program: *Vol. 2 6–19*

---

**X**

XENIX semaphore, maximum parameter: *Vol. 3 4–166*  
XSDSEGS: *Vol. 3 3–38, 4–164*  
XSDSLOTS: *Vol. 3 3–38, 4–165*  
XSEMMAX: *Vol. 3 3–38, 4–166*  
xt terminals, maximum supported parameter: *Vol. 3 4–104*