

File System Administration

2550 Garcia Avenue
Mountain View, CA 94043
U.S.A.



SunSoft
A Sun Microsystems, Inc. Business

© 1994 Sun Microsystems, Inc.
2550 Garcia Avenue, Mountain View, California 94043-1100 U.S.A.

All rights reserved. This product and related documentation are protected by copyright and distributed under licenses restricting its use, copying, distribution, and decompilation. No part of this product or related documentation may be reproduced in any form by any means without prior written authorization of Sun and its licensors, if any.

Portions of this product may be derived from the UNIX[®] and Berkeley 4.3 BSD systems, licensed from UNIX System Laboratories, Inc., a wholly owned subsidiary of Novell, Inc., and the University of California, respectively. Third-party font software in this product is protected by copyright and licensed from Sun's font suppliers.

RESTRICTED RIGHTS LEGEND: Use, duplication, or disclosure by the United States Government is subject to the restrictions set forth in DFARS 252.227-7013 (c)(1)(ii) and FAR 52.227-19.

The product described in this manual may be protected by one or more U.S. patents, foreign patents, or pending applications.

TRADEMARKS

Sun, the Sun logo, Sun Microsystems, Sun Microsystems Computer Corporation, SunSoft, the SunSoft logo, Solaris, SunOS, OpenWindows, DeskSet, ONC, ONC+, and NFS are trademarks or registered trademarks of Sun Microsystems, Inc. in the U.S. and certain other countries. UNIX is a registered trademark of Novell, Inc., in the United States and other countries; X/Open Company, Ltd., is the exclusive licensor of such trademark. OPEN LOOK[®] is a registered trademark of Novell, Inc. PostScript and Display PostScript are trademarks of Adobe Systems, Inc. Archive is a registered trademark of Archive Corporation. Emulex is a registered trademark of Emulex Corporation. Sysgen is a registered trademark of System, Inc. Xylogics is a registered trademark of Xylogics, Inc. All other product names mentioned herein are the trademarks of their respective owners.

All SPARC trademarks, including the SCD Compliant Logo, are trademarks or registered trademarks of SPARC International, Inc. SPARCstation, SPARCserver, SPARCengine, SPARCstorage, SPARCware, SPARCcenter, SPARCclassic, SPARCcluster, SPARCdesign, SPARC811, SPARCprinter, UltraSPARC, microSPARC, SPARCworks, and SPARCcompiler are licensed exclusively to Sun Microsystems, Inc. Products bearing SPARC trademarks are based upon an architecture developed by Sun Microsystems, Inc.

The OPEN LOOK and Sun[™] Graphical User Interfaces were developed by Sun Microsystems, Inc. for its users and licensees. Sun acknowledges the pioneering efforts of Xerox in researching and developing the concept of visual or graphical user interfaces for the computer industry. Sun holds a non-exclusive license from Xerox to the Xerox Graphical User Interface, which license also covers Sun's licensees who implement OPEN LOOK GUIs and otherwise comply with Sun's written license agreements.

X Window System is a product of the Massachusetts Institute of Technology.

THIS PUBLICATION IS PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT.

THIS PUBLICATION COULD INCLUDE TECHNICAL INACCURACIES OR TYPOGRAPHICAL ERRORS. CHANGES ARE PERIODICALLY ADDED TO THE INFORMATION HEREIN; THESE CHANGES WILL BE INCORPORATED IN NEW EDITIONS OF THE PUBLICATION. SUN MICROSYSTEMS, INC. MAY MAKE IMPROVEMENTS AND/OR CHANGES IN THE PRODUCT(S) AND/OR THE PROGRAM(S) DESCRIBED IN THIS PUBLICATION AT ANY TIME.



Contents

Preface.....	xix
<i>Part 1 —Administering File Systems</i>	
1. Planning File Systems	1
Types of File Systems.....	2
Disk-Based File Systems	3
Network-Based File Systems	3
Pseudo File Systems	4
File System Administration Commands.....	7
Syntax of Generic Commands	8
Manual Pages for Generic and Specific Commands	9
How the File System Commands Determine the File System Type.....	9
How to Find Out the Type of a File System	9
The Default SunOS File Systems	12
Understanding Disk Device Names	13
Understanding the UFS File System	13

Disk Slices	14
Cylinder Groups	19
Planning UFS File Systems	19
File System Parameters	20
Making File Systems Available	21
Understanding Mounting and Unmounting	22
The Automounter	29
Sharing Files From a Server	29
2. Creating File Systems	35
Creating a UFS File System on a Disk Slice	35
Preparing to Create a UFS File System	36
Installing a Boot Block on a UFS File System	37
Creating a File System on a Diskette	37
Creating a Temporary File System (TMPFS)	38
Creating a Loopback File System	39
3. Mounting and Unmounting File Systems	41
Finding Out Which File Systems Are Mounted	41
Creating Entries in the File System Table	42
Mounting File Systems in the File System Table	45
Mounting File Systems From the Command Line	47
Unmounting File Systems	53
Sharing File Systems	55
Examples of Automatic Sharing Entries in the /etc/dfs/dfstab File	56

4. Copying UFS Files and File Systems	57
Commands for Copying File Systems	58
Copying Complete File Systems	58
Using the <code>labelit</code> and <code>volcopy</code> Commands	59
Using the <code>dd</code> Command	63
Using the <code>cpio</code> Command	64
Copying Files and File Systems to Tape	65
Useful Commands for Streaming Tape Cartridges	65
Using the <code>tar</code> Command	67
Using the <code>cpio</code> Command	69
Copying Files and File Systems to Diskette	72
Copying Files with a Different Header Format	76
Retrieving Files Created With the <code>bar</code> Command	77
5. The Cache File System	79
Cache File System Concepts	79
Resource Management	80
Front File System Requirements	80
Setting Up a Cached File System	80
Administering a Cache	81
Creating a Cache	81
Cache Parameters	81
Setting Cache Parameters	83
Modifying Parameters	84
Displaying Information About Cached File Systems	85

Deleting Cached File Systems	86
Mounting a Cached File System.	86
Using <code>mount</code> to Mount a Cached File System	87
Creating a <code>vfstab</code> Entry for Cached File Systems	91
Checking a Cached File System	91
<i>Part 2 —Backup and Restore</i>	
6. Planning a Backup Strategy	95
Why You Back Up File Systems	95
Understanding the <code>ufsdump</code> Command	96
Advantages of <code>ufsdump</code>	96
Disadvantages of <code>ufsdump</code>	97
How <code>ufsdump</code> Works	97
Choosing Which File Systems to Back Up	99
File Systems to Back Up on a Standalone System	100
File Systems to Back Up on a Server	100
Choosing Which Media to Use.	102
Backup Device Names.	103
Guidelines for Drive Maintenance and Media Handling . .	107
Considering Other Issues	108
When to Run Backups.	108
How Long to Save Backups	109
How to Back Up Files to a Remote Drive	109
Do You Need to Become Superuser?	110
Should You Check File Systems Before a Full Backup? . . .	110

Do You Want to Put Multiple Backups on the Same Tape?	111
Where Do the Files Reside?	111
How Do You Backup on a Heterogeneous Network?	111
What Are the Security Issues?	112
Planning a Backup Schedule	112
7. Backing Up Files and File Systems	121
Preparing to Do Backups	121
Doing Complete Backups	126
Doing Incremental Backups	128
Backing Up Individual Files and Directories	128
Using a Remote Drive to Do Backups	128
Doing Backups on Remote Systems	130
Troubleshooting When the Root File System Fills Up.	130
Options and Arguments for the <code>ufsdump</code> Command.	131
Default Command Options	132
Options for the <code>ufsdump</code> Command.	132
8. Restoring Files and File Systems	139
Preparing to Restore Files and File Systems.	139
Determining Which Tapes to Use	140
Determining the Disk Device Name	141
Determining the Type of Tape Drive You Will Use	142
Determining the Tape Device Name	142
Restoring Complete File Systems.	142
Synopsis of Steps	142

Restoring Individual Files and Directories	146
Using a Remote Drive to Restore Files	150
Troubleshooting	150
Make Sure the Backup and Restore Commands Match . . .	150
Check to Make Sure You Have the Right Current Directory	151
Use the Old <code>restore</code> Command to Restore Multivolume Diskette Backups	151
Options and Arguments for the <code>ufsrestore</code> Command	151
Command Syntax	151
Options and Arguments	152

Part 3 —Configuring Swap Space and Managing Disk Use

9. Configuring Additional Swap Space	159
About Swap Space	159
Swap Space and Virtual Memory	160
Swap Space and the <code>tmpfs</code> File System	161
How Do I Know If I Need More Swap Space?	162
Swap-Related Error Messages	162
<code>tmpfs</code> -Related Error Messages	162
How Swap Space is Allocated	163
The <code>/etc/vfstab</code> File	163
Planning for Swap Space	164
Monitoring Swap Resources	165
Adding More Swap Space	166
Creating a Swap File	166

Removing a Swap File From Use	168
10. Managing Disk Use	171
Monitoring Available Disk Space	171
Using the <code>df</code> Command	172
Monitoring Files and Directories	176
Monitoring System Log Files That Grow	176
Finding Large Files	178
Finding Large Directories	180
Finding Large Space Users	182
Finding Old and Inactive Files	185
Reducing Overloaded File Systems	186
Truncating Files That Grow	186
Deleting Old or Inactive Files	187
Clearing Out Temporary and Obsolete Files	188
Deleting <code>core</code> Files	189
Removing Crash Dump Files	190
Creating Links Instead of Duplicating Files	191
Moving Directory Trees Between File Systems	192
Controlling Disk Space With Quotas	193
Setting Up and Administering Quotas	194
How Quotas Affect Users	195
How Disk Quotas Work	195
Administering Disk Quotas	201

Part 4 — Troubleshooting

11. Recognizing File Access Problems.	209
Recognizing Problems With Search Paths.	209
Recognizing Problems With Permission and Ownership.	212
Recognizing Problems With Network Access	215
12. Checking File System Integrity	217
Understanding How the File System State Is Recorded	218
What <code>fsck</code> Checks and Tries to Repair.	219
Why Inconsistencies May Occur	219
The UFS Components That Are Checked for Consistency.	220
Error Messages.	226
General <code>fsck</code> Error Messages	227
Initialization Phase <code>fsck</code> Messages	229
Phase 1: Check Blocks and Sizes Messages	232
Phase 1B: Rescan for More DUPs Messages.	236
Phase 2: Check Path Names	236
Phase 3: Check Connectivity Messages.	244
Phase 4: Check Reference Counts Messages.	247
Phase 5: Check Cylinder Groups Messages	251
Cleanup Phase Messages	252
Modifying Automatic Boot Checking	253
The <code>/etc/vfstab</code> File	253
Interactively Checking and Repairing a UFS File System	255
Preening UFS File Systems	257
Restoring a Bad Superblock	258

How to Fix a UFS File System <code>fsck</code> Cannot Repair	260
Syntax and Options for the <code>fsck</code> Command	260
Generic <code>fsck</code> Command Syntax, Options, and Arguments	261
A. File System Reference	265
Default Directories for <code>/</code> and <code>/usr</code> File Systems	265
The Structure of UFS File System Cylinder Groups	270
The Boot Block	271
The Superblock	271
Inodes	272
Storage Blocks	273
Free Blocks	273
Deciding on Custom File System Parameters	274
Logical Block Size	275
Fragment Size	275
Minimum Free Space	276
Rotational Delay (Gap)	277
Optimization Type	277
Number of Bytes per Inode	278
Commands for Creating a Customized File System	278
The <code>newfs</code> Command Syntax, Options, and Arguments . .	278
The Generic <code>mkfs</code> Command	281
B. Bibliography	283
Glossary	285
Index	295

Tables

Table P-1	Typographic Conventions	xxiii
Table 1-1	Generic File System Administrative Commands	7
Table 1-2	The Default SunOS File Systems	12
Table 1-3	Default Parameters Used by the <code>newfs</code> Command	20
Table 1-4	Commands for Mounting and Unmounting File Systems	28
Table 3-1	CD-ROM and Floppy File System Mount Points	44
Table 3-2	CD-ROM and Floppy Device Locations in <code>/vol1</code> —No File System Present	44
Table 3-3	UFS-Specific Mount Options	49
Table 3-4	NFS-Specific Mount Options	50
Table 3-5	CD-ROM Mount Points	51
Table 3-6	Diskette Mount Points	52
Table 3-7	Codes Used by the <code>fuser</code> Command	53
Table 4-1	Summary of Tasks and Available Commands	58
Table 5-1	CacheFS Parameters	82
Table 5-2	Default Values of Cache Parameters	82
Table 5-3	Arguments Used With the <code>-o mount</code> Option	89

Table 6-1	Default File Systems on a Standalone System	100
Table 6-2	Default File Systems on a Server	101
Table 6-3	Media Storage Capacities	102
Table 6-4	Basic Device Names for Backup Devices	103
Table 6-5	Device Abbreviations for Tape Controllers/Units and Media	105
Table 6-6	Designating Density for Rack-Mounted 1/2-inch Tape Drives	106
Table 6-7	Designating Format or Density for SCSI Tape Drives	107
Table 6-8	Daily Cumulative/Weekly Cumulative Backup Schedule . . .	113
Table 6-9	Contents of Tapes for Daily/Weekly Cumulative Schedule .	113
Table 6-10	Daily Cumulative/Weekly Incremental Backup Schedule . . .	114
Table 6-11	Contents of Tapes for Daily Cumulative/Weekly Incremental Schedule	114
Table 6-12	Daily Incremental/Weekly Cumulative Backup Schedule . . .	115
Table 6-13	Contents of Tapes for Daily/Weekly Cumulative Schedule .	115
Table 6-14	Schedule of Backups for an Example Server	117
Table 7-1	Options for the <code>ufsdump</code> Command	132
Table 7-2	Arguments to <code>ufsdump</code> to Specify Tape Capacity	137
Table 8-1	One Required Option for the <code>ufsrestore</code> Command	152
Table 8-2	Additional Options for the <code>ufsrestore</code> Command	153
Table 8-3	Commands for Interactive Restore	154
Table 9-1	Default Swap Space Allocations	163
Table 9-2	Output of the <code>swap -s</code> Command	166
Table 9-3	Options to the <code>mkfile</code> Command	167
Table 10-1	Options for the <code>df</code> Command	172
Table 10-2	Commands Used to Monitor Files and Directories	176
Table 10-3	System Log Files That Grow	176

Table 10-4	Options for the <code>du</code> Command	180
Table 10-5	Disk Quota Commands	194
Table 11-1	Octal Values for File Permissions	214
Table 12-1	State Flag Transitions After <code>fsck</code>	219
Table 12-2	Abbreviations Used in <code>fsck</code> Messages	227
Table A-1	Default Directories for <code>root</code> and <code>/usr</code> File Systems	265

Figures

Figure 1-1	How a Disk Slice Is Formatted for a UFS File System	14
Figure 1-2	A File System	23
Figure 1-3	Mounting a Home Directory File System	24
Figure 6-1	Tape Drive Device Names	104
Figure 10-1	Disk Quotas in Operation.	197
Figure A-1	The File System Address Chain in a UFS System.	273
Figure A-2	A Typical UFS File System.	274

Preface

File System Administration contains conceptual descriptions and procedures for administering files and file systems, backing up and restoring file systems, adding swap space, managing disk use, and troubleshooting in the SunOS™ operating system.

Some Words About Task Orientation

This book contains clear, easy-to-follow steps for each task you need to perform to set up and administer file systems, add additional swap space, and manage disk use. Each set of steps is usually followed by an example that shows you what to type for each step in the example, and the resulting messages, if any, that the system displays.

Descriptions of the underlying concepts of file systems and backup strategies appear in separate chapters. Understanding how these services work can help you troubleshoot problems. If you are familiar with file system concepts and have planned your backup strategy, you can turn directly to the chapters that describe administrative procedures.

Who Should Use This Book

This book is written for system administrators who have a basic working knowledge of SunOS system software, and who are familiar with windowing environments and mouse- and menu-driven applications.

Other Books You Need to Use

- If you need to add a disk, see *Peripherals Administration*.
- If you need to format or partition disks, see *Peripherals Administration*.
- If you need information about security, see *Security, Performance, and Accounting Administration*.
- If you need to set up printers or mail, or administer users and groups, see *User Accounts, Printers, and Mail Administration*.
- If you need detailed information about using Administration Tool, see the *Administration Application Reference Manual*.

How This Book Is Organized

This book is organized into 4 parts, 12 chapters, and 2 appendixes.

Part 1 – “Administering File Systems”

Chapter 1, “Planning File Systems,” explains the types of file systems, describes the default SunOS 5.x file system, disk device names and the UFS file system. It also provides information for planning UFS file systems and for making file systems available.

Chapter 2, “Creating File Systems,” provides steps to create UFS file systems on a disk partition, file systems on a diskette, temporary file systems, and loopback file systems.

Chapter 3, “Mounting and Unmounting File Systems,” provides steps to find out which file systems are mounted, to create entries in the file system table, to mount and unmount file systems, and to make file systems available for NFS mounting from a server.

Chapter 4, “Copying UFS Files and File Systems,” provides steps for copying files and file systems onto removable media such as tapes or diskettes, or onto other systems.

Chapter 5, “The Cache File System,” describes how to create and maintain cached file systems.

Part 2 – “Backup and Restore”

Chapter 6, “Planning a Backup Strategy,” describes why you need a back up strategy, explains the `ufsdump` command and how it works, and describes how to choose which file systems to back up, which media to use, and how to plan a backup schedule.

Chapter 7, “Backing Up Files and File Systems,” provides steps for using the `ufsdump` command to backup files and file systems.

Chapter 8, “Restoring Files and File Systems,” provides steps for using the `ufsrestore` command to restore files and file systems.

Part 3 – “Configuring Swap Space and Managing Disk Use”

Chapter 9, “Configuring Additional Swap Space,” tells you how to add additional swap space without reconfiguring a disk.

Chapter 10, “Managing Disk Use,” explains how to monitor disk use, manage disk quotas, and monitor and remove large files. It also provides a checklist of what to do if a disk fails.

Part 4 – “Troubleshooting”

Chapter 11, “Recognizing File Access Problems,” explains how to recognize and repair common problems users encounter when trying to access files.

Chapter 12, “Checking File System Integrity,” explains the `fsck` file system check program and how to use it.

Appendix A, “File System Reference,” explains the directory and file structure of the default SunOS file system, the structure of UFS file system disk cylinders, and the options you can use for the `newfs` command if you choose not to use the default values.

Appendix B, “Bibliography,” provides a list of published books on system administration that you can use to supplement the information in the system administration documentation set.

Glossary is a list of words and phrases found in this book, and their definitions.

Related Books

For information about basic operating system commands and shells, see the *Solaris Advanced User's Guide*.

These books in the system administration documentation set contain information related to the tasks described in this book. Cross-references in the text refer you to the appropriate book.

- *TCP/IP Network Administration Guide*
- *Name Services Administration Guide*
- *NFS Administration Guide*
- *Peripherals Administration*
- *User Accounts, Printers, and Mail Administration*
- *Security, Performance, and Accounting Administration*

What Typographic Changes and Symbols Mean

Table P-1 describes the typographic conventions used in this book.

Table P-1 Typographic Conventions

Typeface or Symbol	Meaning	Example
AaBbCc123	The names of commands, files, and directories; on-screen computer output	Edit your <code>.login</code> file. Use <code>ls -a</code> to list all files. system% You have mail
AaBbCc123	What you type, contrasted with on-screen computer output	system% su password:
<i>AaBbCc123</i>	Command-line “placeholder” (variable); replace with a real name or value	To delete a file, type <code>rm filename</code> .
<i>AaBbCc123</i>	Book titles, new words or terms, or words to be emphasized	Read Chapter 6 in <i>User’s Guide</i> . These are called <i>class</i> options. You <i>must</i> be root to do this.
%	UNIX C shell prompt	system%
\$	UNIX Bourne or Korn shell prompt	\$
#	Superuser prompt, Bourne or Korn shell	#
#	Superuser prompt, C shell	system#

For steps, press Return only when instructed to do so, even if the text breaks at the end of the line, as shown in this example:

1. **Type** `pmadm -a -p tcp -s lpd -i root -m 'nlsadmin -o /var/spool/lp/fifos/listenBSD -A '\xuniversal-address' -v 'nlsadmin -V'` **and press Return.**

Examples in code boxes that have a backslash (\) at the end of a line are continued onto the next line. The backslash is not part of the example. If the line is an example of what to type, press Return at the end of a line that does not end with a backslash.

```
# pmadm -a -p tcp -s lpd -i root -m `nlsadmin -o \  
/var/spool/lp/fifos/listenBSD -A \  
'\x000202038194180e0000000000000000' -v `nlsadmin -V`
```

When following steps or using examples, be sure to type double-quotes ("), left single-quotes ('), and right single-quotes (') exactly as shown.

Because we assume that the root path will include the `/sbin`, `/usr/sbin`, `/usr/bin`, and `/etc` directories, the steps show the commands in these directories without absolute path names. Steps that use commands in other, less common, directories show the absolute path in the example.

The examples in this book are for a basic SunOS 5.x software installation without the Binary Compatibility Package installed and without `/usr/ucb` in the path.

Caution – If `/usr/ucb` is included in a search path, it should always be at the end. Commands like `ps` or `df` are duplicated in `/usr/ucb` with different formats and options from the SunOS 5.x commands.

Because the Solaris system software provides the Bourne (default), Korn, and C shells, examples in this book show prompts for each of the shells. The C shell prompt is `system-name%`. The Bourne and Korn shell prompt is `$`. The root prompt for all shells is shown by a pound sign (`#`). In examples that affect more than one system, the C shell prompt (which shows the system name) is used to make it clearer when you change from one system to another.

Part 1 — Administering File Systems

This part has the following five chapters:

Chapter 1, “Planning File Systems,” describes the types of file systems, the default Solaris 2.3 file system, disk device names, and the UFS file system. It also provides information for planning UFS file systems and for making file systems available.

Chapter 2, “Creating File Systems,” provides steps to create UFS file systems on a disk partition, file systems on a floppy diskette, temporary file systems, and loopback file systems.

Chapter 3, “Mounting and Unmounting File Systems,” provides steps to find out which file systems are mounted, create entries in the file system table, mount and unmount file systems, and make file systems from a server available for NFS mounting.

Chapter 4, “Copying UFS Files and File Systems,” provides steps for copying UFS files and file systems onto removable media such as tapes or diskettes, or onto other systems.

Chapter 5, “The Cache File System,” describes how to create, modify, and mount cached file systems.

Planning File Systems

This chapter contains these sections:

<i>Types of File Systems</i>	<i>page 2</i>
<i>File System Administration Commands</i>	<i>page 7</i>
<i>The Default SunOS File Systems</i>	<i>page 12</i>
<i>Understanding Disk Device Names</i>	<i>page 13</i>
<i>Understanding the UFS File System</i>	<i>page 13</i>
<i>Planning UFS File Systems</i>	<i>page 19</i>
<i>Making File Systems Available</i>	<i>page 21</i>

A *file system* is a structure of directories used to locate and store files. The term “file system” is used in several different ways:

- To describe the entire file tree from the root directory downward
- To describe a particular type of file system: disk-based, network-based, or pseudo
- To describe the data structure of a disk slice or other media storage device
- To describe a portion of a file tree structure that is attached to a mount point on the main file tree so that it is accessible

Usually, you can tell from context which meaning is intended.

The Solaris system software uses the virtual file system (VFS) architecture, which provides a standard interface for different file system types. The kernel handles basic operations, such as reading, writing, and listing files, without requiring the user or program to know about the underlying file system type.

The file system administrative commands provide a common interface that allows you to maintain file systems of differing types. These commands have two components: a generic component and a component specific to each type of file system. The generic commands apply to most types of file systems, while the specific commands apply to only one type of file system.

Administering the file system is one of your most important system administration tasks. The file system story is a complex one, and understanding it can help you more effectively administer file systems. Read this chapter for background and planning information. Refer to other chapters in this book or to other books in the System Administration book set for instructions about these tasks:

- Making local and remote files available to users. See Chapter 3, “Mounting and Unmounting File Systems,” for detailed information.
- Connecting and configuring new storage devices when needed. See *Peripherals Administration* for detailed information.
- Designing and implementing a backup schedule and restoring files and file systems as needed. See Chapter 6, “Planning a Backup Strategy,” for information on designing a backup schedule. See Chapter 7, “Backing Up Files and File Systems,” for detailed information about doing backups. See Chapter 8, “Restoring Files and File Systems,” for detailed information about restoring files and file systems.
- Checking for and correcting file system damage. File systems are usually checked and corrected at boot time. See Chapter 12, “Checking File System Integrity,” for detailed information on how to proceed if the automatic checking fails.

Types of File Systems

The system software supports three types of file systems:

- Disk-based
- Network-based
- Pseudo

Disk-Based File Systems

Disk-based file systems are stored on physical media such as hard disks, CD-ROMs, and diskettes. Disk-based file systems can be written in different formats. The available formats are:

- UFS – UNIX® file system (based on the BSD Fat Fast File system that was provided in the 4.3 Tahoe release). UFS is the default disk-based file system in SunOS system software.
- HSFS – High Sierra and ISO 9660 file system. High Sierra is the first CD-ROM file system; ISO 9660 is the official standard version of the High Sierra File System. The HSFS file system is used on CD-ROM, and is a read-only file system. SunOS HSFS supports Rock Ridge extensions to ISO 9660, which, when present on a CD-ROM, provide all UFS file system semantics and file types except for writability and hard links.
- PCFS – PC file system, which allows read/write access to data and programs on DOS-formatted disks written for DOS-based personal computers.

Each type of disk-based file system is customarily associated with a particular media device:

- UFS with hard disk
- HSFS with CD-ROM
- PCFS with diskette
- s5 with hard disk

These associations are not, however, restrictive. For example, CD-ROMs and diskettes can have UFS file systems put on them.

Network-Based File Systems

Network-based file systems are file systems accessed over the network. Typically, network-based file systems are file systems that reside on one system and are accessed by other systems across the network. NFS is the only available network-based file system.

NFS is the default SunOS distributed file system. You administer distributed file systems by exporting them from a server and mounting them on individual systems. See “Making File Systems Available” on page 21 for more information.

Pseudo File Systems

Pseudo file systems are virtual or memory-based file systems that provide access to special kernel information and facilities. Most pseudo file systems do not use file system disk space. Cache File Systems use a file system to contain the cache. Some pseudo file systems, such as the temporary file system, may use the swap space on a physical disk.

The Cache File System (CacheFS)

The Cache File System can be used to improve performance of remote file systems or slow devices such as CD-ROM. When a file system is cached, the data read from the remote file system or CD-ROM is stored in a cache on the local system. See Chapter 5, “The Cache File System,” for detailed information on setting up and administering Cache File Systems.

The Temporary File System (TMPFS)

The TMPFS file system uses local memory for file system reads and writes. Access to files in a TMPFS file system is typically much faster than to files in a UFS file system. Files in the temporary file system are not permanent. They are deleted when the file system is unmounted and when the system is shut down or rebooted.

TMPFS is the default file system type for the `/tmp` directory in the SunOS system software. You can copy or move files into or out of the `/tmp` directory, just as you would in a UFS `/tmp` file system.

Using TMPFS file systems can improve system performance by saving the cost of reading and writing temporary files to a local disk or across the network. For example, temporary files are created when you compile a program. The operating system generates a lot of disk or network activity while manipulating these files. Using TMPFS to hold these temporary files may significantly speed up their creation, manipulation, and deletion.

The TMPFS file system uses swap space as a temporary backing store. If a system with a TMPFS file system does not have adequate swap space, two problems can occur:

- The TMPFS file system can run out of space, just as a regular file system can fill up.

- Because TMPFS allocates swap space to save file data (if necessary), some programs may not be able to execute because there is not enough swap space.

See Chapter 2, “Creating File Systems,” for information on how to create TMPFS file systems. See Chapter 9, “Configuring Additional Swap Space,” for information about increasing swap space.

The Loopback File System (LOFS)

The LOFS file system lets you create a new virtual file system. You can access files using an alternative path name. For example, you can create a loopback mount of `/` onto `/tmp/newroot`. The entire file system hierarchy looks like it is duplicated under `/tmp/newroot`, including any file systems mounted from NFS servers. All files are accessible either with a path name starting from `/`, or with a path name starting from `/tmp/newroot`.

See Chapter 2, “Creating File Systems,” for information on how to create LOFS file systems.

The Process File System (PROCFS)

The PROCFS file system resides in memory. It contains a list of active processes, by process number, in the `/proc` directory. Information in the `/proc` directory is used by commands like `ps`. Debuggers and other

development tools can also access the address space of the processes using file system calls. The following example shows a partial listing of the contents of the `/proc` directory:

```
jupiter% ls -l /proc
total 144944
-rw----- 1 root    root          0 Dec 19 15:45 00000
-rw----- 1 root    root    196608 Dec 19 15:45 00001
-rw----- 1 root    root          0 Dec 19 15:45 00002
-rw----- 1 root    root   1028096 Dec 19 15:46 00073
-rw----- 1 root    root   1445888 Dec 19 15:46 00091
-rw----- 1 root    root   1142784 Dec 19 15:46 00093
-rw----- 1 root    root   1142784 Dec 19 15:46 00095
-rw----- 1 ignatz  staff   1576960 Dec 19 15:50 00226
-rw----- 1 ignatz  staff   192512 Dec 19 15:51 00236
-rw----- 1 ignatz  staff  1269760 Dec 19 15:52 00240
-rw----- 1 ignatz  staff  6090752 Dec 19 15:52 00241
-rw----- 1 ignatz  staff   188416 Dec 19 15:52 00247
-rw----- 1 ignatz  staff  2744320 Dec 19 15:52 00256
```



Caution – Do not delete the files in the `/proc` directory. Deleting processes from the `/proc` directory is not the recommended way to kill them. Remember, `/proc` files do not use disk space, so there is little reason to delete files from this directory.

The `/proc` directory does not require system administration.

Additional Pseudo File Systems

These additional types of pseudo file systems are listed for your information. They do not require administration.

- FIFOFS (first-in first-out) – Named pipe files that give processes common access to data
- FDFS (file descriptors) – Provides explicit names for opening files using file descriptors
- NAMEFS – Used mostly by STREAMS for dynamic mounts of file descriptors on top of files

- SPECFS (special) – Provides access to character special and block devices
- SWAPFS – File system used by the kernel for swapping

File System Administration Commands

Most file system administration commands have a *generic* and a *file system-specific* component. Use the generic commands, which call the file system-specific component. Table 1-1 lists the generic file system administrative commands, which are located in the `/usr/bin` directory.

Table 1-1 Generic File System Administrative Commands

Command	Description
<code>clri(1M)</code>	Clears inodes
<code>df(1M)</code>	Reports the number of free disk blocks and files
<code>ff(1M)</code>	Lists file names and statistics for a file system
<code>fsck(1M)</code>	Checks the integrity of a file system and repairs any damage found
<code>fsdb(1M)</code>	File system debugger
<code>fstyp(1M)</code>	Determines the file system type
<code>labelit(1M)</code>	Lists or provides labels for file systems when copied to tape (for use by the <code>volcopy</code> command only)
<code>mkfs(1M)</code>	Makes a new file system
<code>mount(1M)</code>	Mounts file systems and remote resources
<code>mountall(1M)</code>	Mounts all file systems specified in a file system table
<code>ncheck(1M)</code>	Generates a list of path names with their i-numbers
<code>umount(1M)</code>	Unmounts file systems and remote resources
<code>umountall(1M)</code>	Unmounts all file systems specified in a file system table
<code>volcopy(1M)</code>	Makes an image copy of a file system

Most of these commands also have a file system-specific counterpart.



Caution – If you specify an operation on a file system that does not support it, the generic command displays this error message:

command: Operation not applicable for `FSType type`

Syntax of Generic Commands

Most of these commands use this syntax:

```
command [-F type] [-V] [generic-options] [-o specific-options] [special|mount-point]  
[operands]
```

The options and arguments to the generic commands are:

-F *type*

Specifies the type of file system. If you do not use this option, the command looks for an entry which matches *special* or *mount point* in the `/etc/vfstab` file. Otherwise, the default is taken from the file `/etc/default/fs` for local file systems and from the file `/etc/dfs/fstypes` for remote file systems.

-V

Echoes the completed command line. The echoed line may include additional information derived from `/etc/vfstab`. Use this option to verify and validate the command line. The command is not executed.

generic-options

Options common to different types of file systems.

-o *specific-options*

A list of options specific to the type of file system. The list must have the following format: `-o` followed by a space, followed by a series of *keyword* [=value] pairs separated by commas with no intervening spaces.

special|*mount-point*

Identifies the file system. The name must be either the mount point or the special device file for the slice holding the file system. For some commands, the *special* file must be the raw (character) device and for other commands it must be the block device. See “Understanding Disk Device Names” on page 13 for more information about disk device names. In some cases, this argument is used as a key to search the file `/etc/vfstab` for a matching entry from which to obtain other information. In most cases, this argument is required and must come immediately after *specific-options*. However, it is not required when you want a command to act on all the file systems (optionally limited by type) listed in the `/etc/vfstab` file.

operands

Arguments specific to a type of file system. See the specific manual page of the command (for example, `mkfs_ufs(1M)`) for a detailed description.

Manual Pages for Generic and Specific Commands

Both the generic and specific commands have manual pages in the *man Pages(1M): System Administration Commands*. The specific manual page is a continuation of the generic manual page. To look at a specific manual page, append an underscore and the file system type abbreviation to the generic command name. For example, to see the specific manual page for mounting an HSFS file system, type `man mount_hsfs(1M)`.

How the File System Commands Determine the File System Type

The generic file system commands determine the file system type by following this sequence:

1. From `-F`, if supplied.
2. By matching a special device with an entry in `/etc/vfstab` (if `special` is supplied). For example, `fsck` first looks for a match against the `fsck device` field; if no match is found, it then checks against the `special device` field.
3. By using the default specified in `/etc/default/fs` for local file systems and in `/etc/dfs/fstypes` for remote file systems.

How to Find Out the Type of a File System

You can determine the type of a file system from the same files the generic commands use:

- The `FS type` field in the file system table (`/etc/vfstab`)
- The `/etc/default/fs` file for local file systems
- The `/etc/dfs/fstypes` file for remote file systems

To find a file system's type in the `/etc/vfstab` file:

- ♦ **Type** `grep mount-point /etc/vfstab` **and press Return.** Information for the mount point is displayed.

```
mars% grep /tmp /etc/vfstab
swap          -          /tmp          tmpfs        -          yes          -
mars%
```

If `vfstab` does not have an entry for a file system, use one of the following procedures to determine the file system's type.

To identify a mounted file system's type:

- ◆ **Type `grep mount-point /etc/mnttab` and press Return.**
Information on the mount point is displayed.

```
mars% grep /home /etc/mnttab
mars:(pid129) /home nfs ro,ignore,map=/etc/auto_home,indirect,dev=21c0004 693606637
neptune:/export/home/neptune /tmp_mnt/home/neptune nfs rw,dev=21c0005 695409833
mars%
```

Or

- ◆ **Type `mount` and press Return.**
A list of the mounted file systems and their types is displayed.

```
mars% mount
/ on /dev/dsk/c0t3d0s0 read/write on Tue Dec 24 12:29:22 1993
/usr on /dev/dsk/c0t1d0s6 read/write on Tue Dec 24 12:29:22 1993
/proc on /proc read/write on Tue Dec 24 12:29:22 1991
/usr/man on swsvr4-50:/export/svr4/man read/write/remote on Mon Dec 30 12:49:11 1993
/usr/openwin on swsvr4-50:/export/svr4/openwinV3.jpbl read/write/remote on Mon Dec 30 \
13:50:54 1993
/tmp on swap read/write on Wed Jan 8 13:38:45 1994
/mnt on swsvr4-50:/export/svr4 read/write/remote on Fri Jan 10 15:51:23 1994
/tmp_mnt/home/neptune on neptune:/export/home/neptune read/write/remote on Tue Jan 14 \
09:23:53 1994
mars%
```

Or

1. **Type `devnm mount-point` and press Return.**
The raw device name is displayed.
2. **Become superuser.**

- 3. Type `fstyp /dev/rdisk/cntndnsn` and press Return.**
The type of the file system is displayed.

```
mars% devnm /usr
/dev/dsk/c0t1d0s6 /usr
mars% su
Password:
# fstyp /dev/rdisk/c0t3d0s0
ufs
#
```

The Default SunOS File Systems

The SunOS file system is hierarchical, starting with the `root` directory (`/`) and continuing downwards through a number of directories. The system software installs a default set of directories and uses a set of conventions to group similar types of files together. Table 1-2 describes the default SunOS file systems, and shows the type of each file system.

Table 1-2 The Default SunOS File Systems

File System or Directory	File System Type	Description
<code>/</code>	UFS	The top of the hierarchical file tree. The root directory contains the directories and files critical for system operation, such as the kernel (<code>/kernel/unix</code>), the device drivers, and the programs used to start (boot) the system. It also contains the mount point directories where local and remote file systems can be attached to the file tree.
<code>/usr</code>	UFS	Contains system files and directories that can be shared with other users. Files that run only on certain types of systems are in the <code>/usr</code> directory (for example SPARC® executables). Files (such as manual pages) that can be used on all types of systems are in <code>/usr/share</code> .
<code>/export/home</code> or <code>/home</code>	NFS, UFS	The mount point for users' home directories, which store users' work files. By default <code>/home</code> is an automounted file system. On standalone systems, <code>/home</code> may be a UFS file system on a local disk slice.
<code>/var</code>	UFS	Contains system files and directories that are likely to change or grow over the life of the local system. These include system logs, <code>vi</code> and <code>ex</code> backup files, and <code>uucp</code> files.

Table 1-2 The Default SunOS File Systems (Continued)

File System or Directory	File System Type	Description
/opt	NFS, UFS	Mount point for optional, third-party software. On some systems /opt may be a UFS file system on a local disk slice.
/tmp	TMPFS	Temporary files, cleared each time the system is booted or unmounted.
/proc	PROCFS	Contains a list of active processes, by number.

The root (/) and /usr file systems are both needed to run a system. Some of the most basic commands from the /usr file system (like mount) are included in the root file system so that they are available when the system boots or is in single-user mode and /usr is not mounted. See Appendix A, “File System Reference,” for a complete list of the default directories.

Understanding Disk Device Names

Please refer to the *Administration Supplement for Solaris Platforms* for a description of device naming conventions.

Understanding the UFS File System

UFS is the default disk-based file system in SunOS system software. Most of the time, when you administer a disk-based file system, you will be administering UFS. It provides the following features:

- State flags – Show the state of the file system: clean, stable, active, or unknown. These flags eliminate unnecessary file system checks. If the file system is “clean” or “stable,” fsck is not run.
- Extended fundamental types (EFT) – 32-bit user ID (UID), group ID (GID), and device numbers.
- Large file systems – A UFS file system can be as large as 1 terabyte (1 Tbyte) and can have regular files up to 2 gigabytes (2 Gbytes). SunOS system software does not provide *striping*, which is required to make a logical slice large enough for a 1-Tbyte file system. Products such as Online: DiskSuite™ provide this capability.

The following sections describe how a disk is divided into slices and cylinder groups, and structured as a UFS file system.

Note – SunOS device names use the term *slice* (and the letter “s” in the device name) to refer to the slice number.

Figure 1-1 shows how a disk slice can be formatted to contain a UFS file system.

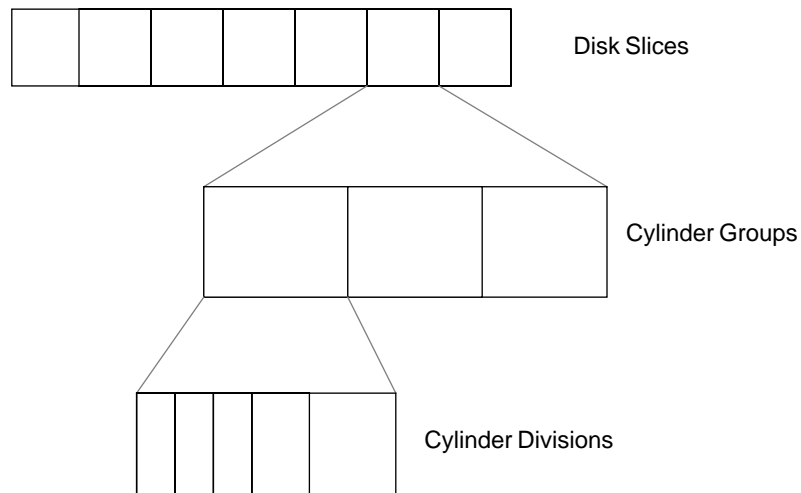


Figure 1-1 How a Disk Slice Is Formatted for a UFS File System

The following sections describe disk slices, cylinder groups, and cylinder divisions.

Disk Slices

A slice is composed of a single range of contiguous blocks, and is a physical subset of the disk (except for slice 2, which customarily represents the entire disk). Before you can create a file system on a disk, you must format and divide it into slices. See *Peripherals Administration* for complete information on installing and formatting disks, and dividing disks into slices.

A slice can be used as a raw device for swap space or to hold one and only one UFS file system. A disk can be divided into as many as eight slices. See the *Administration Supplement for Solaris Platforms* for a list of customary disk slice assignments.

▼ **How to Get Information About Disks and Disk Slices**

1. Become superuser.

2. Type `format` and press Return.

A list of available disks is displayed.

3. Type a number for the disk you want information about.

The `format` menu and the `format>` prompt are displayed.



Caution – Be sure you follow the steps exactly so that you choose the right options from the `format` and `partition` menus. You can cause major system damage and data loss if you inadvertently reformat a disk or choose some other potentially destructive option from these menus. See the *Administration Supplement for Solaris Platforms* manual for information on the `format` command.

4. Type `partition` and press Return.

The `partition` menu and the `partition>` prompt are displayed.

5. Type `print` and press Return.

A list of the slices currently defined for the disk is displayed. Any slices with values of zero are not defined.

6. Type `quit` and press Return.

The `format` menu and the `format>` prompt are displayed.

7. Type `quit` and press Return.

The shell prompt is displayed.

This example shows disk information for `/dev/dsk/c0t3d0`. Note that the `format` command does not show slice information as part of the available disk selections. It shows the controller, target, and disk number for each disk.

Code Example 1-1 Using the format Command (1 of 2)

```
jupiter% su
Password:
# format
AVAILABLE DISK SELECTIONS:
 0. c0t0d0 at scsibus0 slave 24
sd0: <SUN0207 cyl 1254 alt 2 hd 9 sec 36>
 1. c0t3d0 at scsibus0 slave 0: veryloud
sd3: <SUN0207 cyl 1254 alt 2 hd 9 sec 36>
Specify disk (enter its number): 1
FORMAT MENU:
disk - select a disk
type - select (define) a disk type
partition - select (define) a partition table
current - describe the current disk
format - format and analyze the disk
repair - repair a defective sector
label - write label to the disk
analyze - surface analysis
defect - defect list management
backup - search for backup labels
verify - read and display labels
save - save new disk/partition definitions
inquiry - show vendor, product and revision
volname - set 8-character volume name
quit
format> partition
PARTITION MENU:
 0 - change '0' partition
 1 - change '1' partition
 2 - change '2' partition
 3 - change '3' partition
 4 - change '4' partition
 5 - change '5' partition
 6 - change '6' partition
 7 - change '7' partition
select - select a predefined table
modify - modify a predefined partition table
name - name the current table
```

Code Example 1-1 Using the format Command (2 of 2)

```
print - display the current table
label - write partition map and label to the disk
quit
partition> print
Volume: veryloud
Current partition table (original sd3):
Part      Tag      Flag      Cylinders      Size      Blocks
  0       root      wm         0 - 39         14.06MB    (40/0/0)
  1       swap      wu         40 - 199       56.25MB    (160/0/0)
  2      backup      wm         0 - 1150      404.65MB   (1151/0/0)
  3 unassigned      wm           0              0          (0/0/0)
  4 unassigned      wm           0              0          (0/0/0)
  5         -      wm           0          10.20MB    (29/0/0)
  6       usr      wm        200 - 228      121.29MB   (345/0/0)
  7      home      wm        574 - 1150     202.85MB   (577/0/0)
partition> quit
format> quit
#
```

If you know the disk and slice number, you can display information for a disk using the `prtvtoc` (print volume table of contents) command. You can specify the volume by specifying any non-zero-size slice defined on the disk (for example, `/dev/rdisk/c0t3d0s2` for all of disk 3, or `/dev/rdisk/c0t3d0s7` for the eighth slice of disk 3). If you know the target number of the disk, but do not know how it is divided into slices, you can show information for the entire disk by specifying either slice 2 or slice 0.

1. Become superuser.

- 2. Type `prtvtoc /dev/rdisk/cntndnsn` and press Return.**
Information for the disk and slice you specify is displayed.

In this example, information is displayed for all of disk 3:

Code Example 1-2 Using the prtvtoc Command

```
jupiter% su
Password:
# prtvtoc /dev/rdisk/c0t3d0s2
* /dev/rdisk/c0t3d0s2 (volume "") partition map
```

Code Example 1-2 Using the prtvtoc Command (Continued)

```

*
* Dimensions:
*   512 bytes/sector
*   36 sectors/track
*   9 tracks/cylinder
*   324 sectors/cylinder
*   1272 cylinders
*   1254 accessible cylinders
*
* Flags:
*   1: unmountable
*   10: read-only
*
*
* Partition  Tag  Flags      First      Sector      Last
* Partition  Tag  Flags      Sector     Count       Sector  Mount
Directory
      2      5    01          0      406296      406295
      6      4    00          0      242352      242351
      7      0    00      242352      163944      406295  /files7

```

Size Restrictions on UFS File Systems

Total Size

The limit on the total size of a file system is 1 Tbyte. A UFS file system can be as big as the slice that holds it.

Maximum File Size

The maximum size for any one file in a UFS file system is 2 Gbyte.

Maximum Number of Files

The maximum number of files per UFS file system is determined by the number of inodes allocated for a file system. The number of inodes depends on how much disk space is allocated for each inode and the total size of the file system. By default, one inode is allocated for each 2 Kbyte of data space. You can change the default allocation using the `-i` option of the `newfs` command.

Logical Block and Fragment Size

Fragments are small logical blocks that are created to save space by reducing unused portions of logical blocks. The maximum logical block and fragment size is 8192 bytes, although fragments are typically less than or equal to the size of logical blocks.

Cylinder Groups

You create a UFS file system on a disk slice, which is divided into one or more areas called *cylinder groups*. A cylinder group is composed of one or more consecutive disk cylinders (the set of tracks on a group of platters that have the same radial distance from the center of the platter). See *Peripherals Administration* for a complete description of disk geometry.

A *cylinder group map* is created for each cylinder group. The cylinder group map records the block usage and available blocks.

Types of Blocks

Cylinder groups are divided into blocks to control and organize the structure of the files within the cylinder group. Each type of block has a specific function in the file system. A UFS file system has four types of addressable blocks as well as additional information management disk areas. The four types of blocks are:

- Boot block – Used to store information used when booting the system
- Superblock – Used to store much of the information about the file system
- Inode – Used to store all information about a file except its name
- Storage or data block – Used to store data for each file

See Appendix A, “File System Reference,” for more detailed information about each type of block.

Planning UFS File Systems

Once disks are formatted and divided into slices, you need to make a file system on each slice that will contain UFS files. See the *Peripherals Administration* for detailed information on how to format disks and divide them into slices.

When laying out file systems, you need to consider possible conflicting demands. Here are some suggestions:

- Distribute the work load as evenly as possible among different I/O systems and disk drives. Distribute `/home` and `swap` directories evenly across disks.
- Keep projects or groups within the same file system.
- Use as few file systems per disk as possible. On the system (or boot) disk you usually have three slices: `/`, `/usr`, and a swap area. On other disks create one or at most two slices. Fewer, roomier slices cause less file fragmentation than many small, over-crowded slices. Higher-capacity tape drives and the ability of `ufsdump` to handle multiple volumes make it easier to back up larger file systems.
- Most sites do not need to be concerned about keeping similar types of user files in the same file system. Infrequently, you may have some users who consistently create very small or very large files. In such a case, you might consider creating a separate file system with more inodes for users who consistently create very small files.

File System Parameters

To make a new file system on a disk slice, you almost always use the `newfs` command. Table 1-3 shows the default parameters the `newfs` command uses:

Table 1-3 Default Parameters Used by the `newfs` Command

Parameter	Default Value
Block size	8 Kbyte
Fragment size	1 Kbyte
Minimum free space	10%
Rotational delay	Device-dependent
Optimization type	Space
Number of inodes	1 for each 2 Kbyte of disk space

If you want to customize a file system using arguments to the `newfs` command or with the `mkfs` command, see Appendix A, “File System Reference,” for information about altering these parameters.

Making File Systems Available

Once you have created a file system, you need to make it available. You make file systems available by mounting them. A mounted file system is attached to the system directory tree at the specified mount point, and becomes available to the system. The root file system is always mounted. Any other file system can be connected or disconnected from the root file system.

You can mount a local file system in these ways:

- By creating an entry in the `/etc/vfstab` (virtual file system table) file. The `/etc/vfstab` file contains a list of file systems that are automatically mounted when the system is booted to multiuser state. See “The Virtual File System Table” on page 26 for a description of the `/etc/vfstab` file.
- From the command line using the `mount` command.

File systems on disk slices must always be mounted on the server system and shared (exported) before other systems can access them. See “Sharing Files From a Server” on page 29 and Chapter 3, “Mounting and Unmounting File Systems,” for information about sharing file systems. When file systems are shared from a server, a client can mount them as NFS file systems in any of these three ways:

- By adding an entry to the `/etc/vfstab` (virtual file system table) file so that the file system is automatically mounted when the system is booted to multiuser state.
- By using the `autofs` to automatically mount or unmount the file system when a user changes into (`mount`) or out of (`umount`) the directory. See *NFS Administration Guide* for information about `autofs`.
- By using the `mount` command at a command line.

CD-ROMs containing file systems are mounted when the CD-ROM is inserted. Diskettes containing file systems are mounted by running the `volcheck(1)` command.

Understanding Mounting and Unmounting

File systems can be attached to the hierarchy of directories available on a system. This process is called *mounting*. You need to determine whether the file system should be:

- Entered in the `/etc/vfstab` file to be mounted each time the system is booted
- Whether it can be appropriately mounted using `autofs`
- Whether it will be used only temporarily, and can be mounted from the command line

To mount a file system you need:

- To be superuser
- A mount point on the local system. The mount point is a directory to which the mounted file system is attached.
- The resource name of the file system to be mounted (for example, `/usr`)

As a general rule, local disk slices should always be included in the `/etc/vfstab` file. Any software from servers, such as OpenWindows or manual pages, and home directories from a server can be either included in the `/etc/vfstab` file or automounted, depending on the policy at your site.

When you mount a file system, any files or directories that might be present in the mount point directory are unavailable as long as the file system is mounted. These files are not permanently affected by the mounting process, and become available again when the file system is unmounted. However, mount directories are usually empty, because you usually do not want to obscure existing files.

Figure 1-2 shows the root file system with subdirectories `sbin`, `etc`, and `home`:

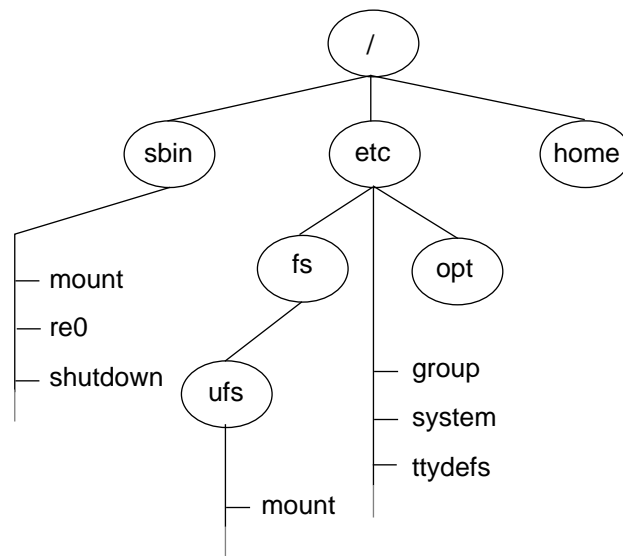


Figure 1-2 A File System

To attach a user's home directory to the empty `/home` directory mount point, first create a directory for the new user. For a user named `ignatz`, create a directory in `/home` named `ignatz`, giving it the appropriate permissions and ownership. Then mount the file system. When the `ignatz` file system is mounted, all of the files and directories in `/home/ignatz` are available, as shown in Figure 1-3. You can also create other user directories in the `/home` directory and use those directories as mount points for other user file systems. See Chapter 3, "Mounting and Unmounting File Systems," for information on how to perform these tasks.

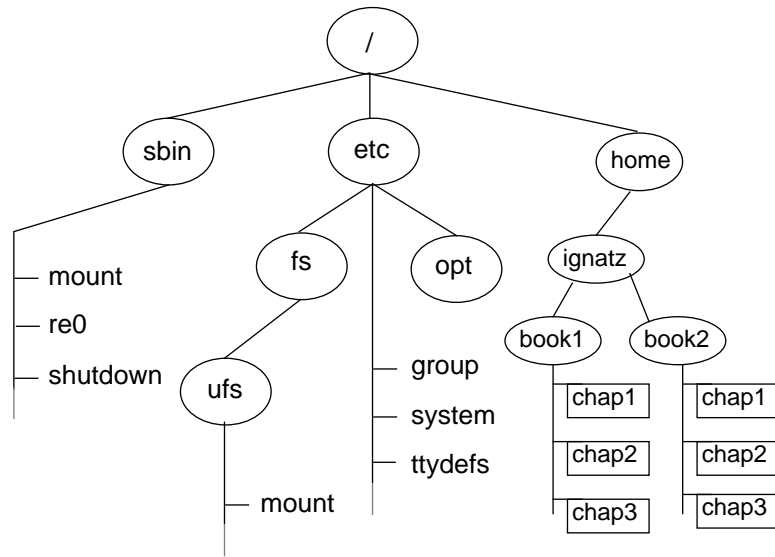


Figure 1-3 Mounting a Home Directory File System

Note - This example illustrates the concept of mounting. Because `/home` is, by default, an `autofs` mount point directory, home directory files would be mounted by `autofs` rather than the `mount` command.

Whenever you mount or unmount a file system, the `/etc/mnttab` (mount table) file is modified to show the list of currently mounted file systems. You can display the contents of the mount table using the `cat` or `more` commands, but you cannot edit it, as you would the `/etc/vfstab` file. Here is an example of a mount table file:

```
mars% more /etc/mnttab
/dev/dsk/c0t3d0s0      /          ufs      rw,suid 693186371
/dev/dsk/c0t1d0s6     /usr       ufs      rw,suid 693186371
/proc /proc proc rw,suid 693186371
swap /tmp tmpfs ,dev=0 693186373
swsvr4-50:/export/svr4/openwinV3.jpbl /usr/openwin nfs rw,dev=21c0000 693186443
swsvr4-50:/export/svr4/man /usr/man nfs rw,dev=21c0001 693186447
mars:(pid127) /nse nfs ro,ignore,map=/etc/auto.nse,indirect,dev=21c0002 693186449
mars:(pid127) /net nfs ro,ignore,map=-hosts,indirect,dev=21c0003 693186449
mars:(pid127) /home nfs ro,ignore,map=/etc/auto_home,indirect,dev=21c0004
693186449
neptune:/export/home/neptune /tmp_mnt/home/neptune nfs rw,dev=21c0005 693186673
mars%
```

Unmounting a file system removes it from the file system mount point, and deletes the entry from the `/etc/mnttab` file. Some file system administration tasks cannot be performed on mounted file systems. You should unmount a file system when:

- It is no longer needed or has been replaced by a file system that contains more current software.
- You check and repair it using the `fsck` command. See Chapter 12, “Checking File System Integrity,” for more information about the `fsck` command.

It is a good idea to unmount a file system before doing a complete backup of it. See Chapter 7, “Backing Up Files and File Systems,” for more information about doing backups.

Note – File systems are automatically unmounted as part of the system shutdown procedure.

The Virtual File System Table

The default file system configuration table (the `/etc/vfstab` file) depends on the selections you make when installing system software. You should edit the `/etc/vfstab` file for each system to automatically mount local UFS file systems, essential NFS file systems, and any other appropriate file systems.

This section describes the `/etc/vfstab` file. See Chapter 3, “Mounting and Unmounting File Systems,” for information on how to edit and use the file. The file system table is an ASCII file. Comment lines begin with `#`. This example shows an `/etc/vfstab` file for a system with two disks and two NFS file systems mounted.

```
# more /etc/vfstab
#device          device          mount          FS      fsck  mount  mount
#to mount        to fsck         point          type    pass  at boot options
/dev/dsk/c0t0d0s0 /dev/rdisk/c0t0d0s0 /      ufs     1      no     -
/proc            -              /proc         proc    -      no     -
/dev/dsk/c0t0d0s1 -              -             swap    -      no     -
swap            -              /tmp          tmpfs   -      yes    -
/dev/dsk/c0t0d0s6 /dev/rdisk/c0t0d0s6 /usr        ufs     2      no     -
/dev/dsk/c0t3d0s7 /dev/rdisk/c0t3d0s7 /files7     ufs     2      no     -
cheers:/export/ivr4/man.ja5 - /usr/man    nfs     -      yes    hard
cheers:/export/ivr4/openwinV3.ja4 - /usr/openwin nfs     -      yes    hard
#
```

Note that, for `/` and `/usr`, the `mount at boot` field value is specified as `no` because these file systems are mounted as part of the boot sequence before the `mountall` command is run. If the `automount` field value is specified as `yes`, the `mountall` program redundantly (and unnecessarily) tries to mount these already mounted file systems. See *Administration Supplement for Solaris Platforms* for a description of the booting procedure.

An entry in the virtual file system table has the following seven fields:

special fsckdev mountp fstype fsckpass mount-at-boot mntopts

The fields are separated by tab or space characters.

Note – You must have an entry in each field in the `/etc/vfstab` file. If there is no value for the field, be sure to enter a dash (-).

special (device to mount)

The device to mount can be:

- The block special device for local UFS file systems (for example, `/dev/dsk/c0t0d0s0`)
- The resource name for remote file systems (for example, `myserver:/export/home` for an NFS file system)
- The name of the slice on which to swap (for example, `/dev/dsk/c0t3d0s1`)
- The `/proc` directory and `proc` file system type

For more information on remote file systems, see *NFS Administration Guide*.

fsckdev (device to fsck)

The raw (character) special device that corresponds to the file system identified by the *special* field (for example, `/dev/rdisk/c0t0d0s0`). This determines the raw interface that is used by `fsck`. Use a dash (-) when there is no applicable device, such as for a read-only file system or a network-based file system.

mountp (mount point)

The default mount point directory (for example, `/usr` for `/dev/dsk/c0t0d0s6`).

fstype (FS type)

The type of file system identified by the *special* field.

fsckpass (fsck pass)

The pass number used by `fsck` to decide whether to check a file system. When the field contains a dash (-), the file system is not checked. When the field contains a value greater than zero, the file system is checked; non-UFS file systems with a zero value for `fsck pass` are checked. For UFS file systems only, when the field contains a zero, the file system is not checked. When `fsck` is run on multiple UFS file systems that have `fsck pass` values greater than one and the `preen` option (`-o p`) is used, `fsck` automatically checks the file systems on different disks in parallel to maximize efficiency. When the field contains a value of 1, the file system is checked sequentially. Otherwise, the value of the pass number does not have any effect.

Note – In SunOS system software, the `fsck pass` field does *not* explicitly specify the order in which file systems are checked.

(mount at boot

`yes` or `no` for whether the file system should be automatically mounted by `mountall` when the system is booted. Note that this field has nothing to do with `autofs`.

mntopts (mount options)

A list of comma-separated options (with no spaces) that are used in mounting the file system. Use a dash (-) to show no options. See the `mount(1M)` manual page for a list of the available options.

Commands Used to Mount and Unmount File Systems

Table 1-4 lists the commands in the `/usr/sbin` directory that you use to mount and unmount file systems.

Table 1-4 Commands for Mounting and Unmounting File Systems

Command	Description
<code>mount(1M)</code>	Mounts file systems and remote resources
<code>mountall(1M)</code>	Mounts all file systems specified in a file system table
<code>umount(1M)</code>	Unmounts file systems and remote resources
<code>umountall(1M)</code>	Unmounts all file systems specified in a file system table

The `mount` commands will not mount a read/write file system that has inconsistencies. If you receive an error message from the `mount` or `mountall` command, you may need to check the file system. See Chapter 12, “Checking File System Integrity,” for information on how to check the file system.

The `umount` commands will not unmount a file system that is busy. A file system is considered busy if a user is in a directory in the file system, or if a program has a file open in that file system. See Chapter 3, “Mounting and Unmounting File Systems,” for more information.

The Automounter

You can mount file systems shared through NFS using a method called *automounting*. `autofs` runs in the background and mounts and unmounts remote directories as they are needed. Whenever a user on a client system running `autofs` accesses a remote file or directory available through `autofs`, it mounts the file system on the user's system. The remote file system remains mounted as long as the user remains in the directory and is using a file. If the remote file system is not accessed for a certain period of time, it is automatically unmounted. `autofs` mounts and unmounts file systems as required without any intervention on the part of the user other than changing into or out of a directory.

You can mount some file hierarchies with `autofs`, and others using the `/etc/vfstab` file and the `mount` command. A diskless machine *must* have entries for `/` (root), `/usr`, and `/usr/kvm` in the `/etc/vfstab` file. Because shared file systems should always remain available, do not use `autofs` to mount `/usr/share`.

The `autofs` works with the file systems specified in *maps*. These maps can be maintained as NIS, NIS+, or local files.

The `autofs` maps can specify several remote locations for a particular file. This way, if one of the servers is down, `autofs` can try to mount from another machine. You can specify which servers are preferred for each resource in the maps by assigning each server a weighting factor.

The `autofs` starts automatically when a system enters run level 3. You can also start it from a command line. See *NFS Administration Guide* for complete information on how to set up and administer `autofs`.

By default, the SunOS system software automounts `/home`.

Sharing Files From a Server

NFS is a distributed file system that can be used to “tie together” computers that are running different operating systems. For example, systems running DOS can share files with systems running UNIX.

NFS makes the actual physical location of the file system irrelevant to the user. You can use NFS to allow users to see all the relevant files, regardless of location. Instead of placing copies of commonly used files on every system,

NFS allows you to place one copy on one system's disk and let all other systems access it across the network. Under NFS, remote file systems are virtually indistinguishable from local ones.

A system becomes an NFS server if it has file systems to *share* or *export* over the network. A server keeps a list of currently exported file systems and their access restrictions (such as read/write or read-only).

You may want to share resources, such as files, directories, or devices from one system on the network (typically, a server) with other systems. For example, you might want to share third-party applications or source files with users on other systems.

When you share a resource, you make it available for mounting by remote systems.

You can share a resource in these ways:

- Using the `share` or `shareall` command
- Adding an entry to the `/etc/dfs/dfstab` (distributed file system table) file

The default `/etc/dfs/dfstab` file shows you the syntax and an example of entries:

```
venus% more /etc/dfs/dfstab

# place share(1M) commands here for automatic execution
# on entering init state 3.
#
# share [-F fstype] [ -o options] [-d "<text>"] <pathname> [resource]
# .e.g,
# share -F nfs -o rw=engineering -d "home dirs" /export/home2
share -F nfs /var/mail

venus%
```

Add one entry to the `/etc/dfs/dfstab` file for each resource that you want to have shared automatically. Each entry must be on a separate line, using this syntax:

```
share [-F nfs] [-o specific-options] [-d "description"] pathname resource
```

-F *nfs*

Indicates that the file system type is NFS. If you have only one distributed file system package installed, *nfs* is the default, and you can omit the -F option.

-o *specific-options*

Is a comma-separated list of options that regulates how the resource is shared. Specific options that can follow the -o flag include:

rw

Shares *pathname* read/write to all clients (by default), except those that are specified under *ro=*.

ro

Shares *pathname* read-only to all clients, except those that are specified under *rw=*.

Note – You cannot specify both *rw* and *ro* without arguments, and you cannot specify the same client in the *rw=* list and the *ro=* list. If no read/write option is specified, the default is read/write for all clients.

ro=client[:client]

Shares *pathname* read-only to the listed client machines or netgroup names (overriding *rw*).

rw=client[:client]

Shares *pathname* read/write to the listed client machines or netgroup names (overriding *ro*).

`anon=uid`

Lets you specify a different *uid* for “anonymous” users—users whose *uid* is 0, the UID of root on Solaris 2.3 systems—when accessing *pathname*. By default, anonymous users are mapped to username `nobody`, which has the UID `UID_NOBODY`. User `nobody` has ordinary user privileges, not superuser privileges.

`root=host[:host]`

Lets a user from host *host*, whose *uid* is 0, access *pathname* as root; root users from all other hosts become `anon`. If this option is not specified, no user from any host is granted access to *pathname* as root.



Caution – Granting root access to other hosts has far-reaching security implications; use the `root=` option with extreme caution.

`secure`

Shares a resource with additional user authentication required (see *Security, Performance, and Accounting Administration* for more information).

`kerberos`

Shares a resource with Kerberos authentication.

`-d description`

Is a comment that describes the resource to be shared. If you use the `-d` option, the description is stored in the `sharetab` file. However, clients do not see the description displayed when they use the `dfshares` command to list the resources shared on that system.

pathname

Is the full name of the resource to be shared, starting at root (`/`).

See Chapter 3, “Mounting and Unmounting File Systems,” for information on how to share files and *NFS Administration Guide* for a complete description of NFS.

Note – Arguments that accept a client or host list (`ro=`, `rw=`, and `root=`) are guaranteed to work over UDP, but may not work over other transport providers.

Under NFS, a server shares resources it owns so clients can mount them. However, a user who becomes the superuser at a client is denied access as the superuser to mounted remote resources. When a user logged in as root on one host requests access to a remote file shared through NFS, the user's ID is changed from 0 to the user ID of the user name `nobody`. The access rights of user `nobody` are the same as those given to the public for a particular file. For example, if the public has only execute permission for a file, then user `nobody` can execute only that file.

See Chapter 3, "Mounting and Unmounting File Systems," for information on how to share files and file systems.

≡ 1

Creating File Systems



This chapter contains these sections:

<i>Creating a UFS File System on a Disk Slice</i>	<i>page 35</i>
<i>Creating a File System on a Diskette</i>	<i>page 37</i>
<i>Creating a Temporary File System (TMPFS)</i>	<i>page 38</i>
<i>Creating a Loopback File System</i>	<i>page 39</i>

Creating a UFS File System on a Disk Slice

You need to create UFS file systems only occasionally. The system software automatically creates file systems as part of the installation process. You need to create (or re-create) a UFS file system when you:

- Add or replace disks
- Change the slices of an existing disk
- Do a full restore on a file system
- Change some parameters (such as block size) of a file system

Use the `newfs` command to create UFS file systems. `newfs` is a convenient front-end to the `mkfs` command, the program that creates the new file system. On Solaris 2.x systems, information used to set some parameter defaults, such as number of tracks per cylinder and number of sectors per track, is read from the disk label. `newfs` determines the file system parameters to use based on the options you choose and information from the disk label, and passes the parameters to the `mkfs` command, which builds the file system.

Although you can use the `mkfs` command directly, you need to know much more about the parameters and how to choose them. `mkfs` supplies defaults for all parameters, but they are not tuned to the underlying hardware. You must also specify the total size of the slice.

Use the `mkfs` command:

- To create a disk-based file system type other than UFS, if the software for such a file system type is available.
- To create a UFS file system whose logical geometry differs from the physical geometry of the disk. For example, you could specify parameters appropriate for a diskette.

See Appendix A, “File System Reference,” for information about choosing parameters and using the `mkfs` command.

Preparing to Create a UFS File System

To create a file system on a formatted disk that has been divided into slices, you need to know the special device file name of the slice that will contain the file system. See the *Administration Supplement for Solaris Platforms* for a description of disk device naming conventions. See “How to Get Information About Disks and Disk Slices” on page 15 for information on finding disks and disk slice numbers.

▼ How to Create a UFS File System

The disk must be formatted and divided into slices before you can create UFS file systems on it. If you are re-creating an existing UFS file system, unmount the file system before performing these steps:

1. **Become superuser.**
2. **Type `newfs /dev/rdisk/device-name` and press Return.**
You are asked if you want to proceed.



Caution – Be sure you have specified the correct device name for the slice before performing the next step. If you specify the wrong slice, you will erase its contents when the new file system is created.

3. Type **y** to confirm.

The `newfs` command uses optimized default values to create the file system.

Example: How to Create a UFS File System

This example creates a file system on `/dev/rdisk/c0t3d0s7`:

```
jupiter% su
Password:
# newfs /dev/rdisk/c0t3d0s7
newfs: construct a new file system /dev/rdisk/c0t3d0s7 (y/n)? y
/dev/rdisk/c0t3d0s7:      163944 sectors in 506 cylinders of 9 tracks, 36 sectors
      83.9MB in 32 cyl groups (16 c/g, 2.65MB/g, 1216 i/g)
super-block backups (for fsck -b #) at:
 32, 5264, 10496, 15728, 20960, 26192, 31424, 36656, 41888,
 47120, 52352, 57584, 62816, 68048, 73280, 78512, 82976, 88208,
 93440, 98672, 103904, 109136, 114368, 119600, 124832, 130064, 135296,
140528, 145760, 150992, 156224, 161456,
#
```

Installing a Boot Block on a UFS File System

See the *Administration Supplement for Solaris Platforms* for information on installing a boot block on a UFS file system.

Creating a File System on a Diskette

See the *Administration Supplement for Solaris Platforms* for information on creating file systems on diskettes.

Creating a Temporary File System (TMPFS)

By default, the `/tmp` directory for the SunOS system software is a TMPFS file system, and an entry is provided for it in the default `/etc/vfstab` file.

The most common use for the TMPFS file system is for the `/tmp` directory, although you may find other ways to use it. If you do create multiple TMPFS file systems, be aware that they all use the same system resources. Files created under one TMPFS directory use up the space available for any other TMPFS file system.

Because files in TMPFS directories do not survive across reboots or unmounts, do not mount TMPFS file systems under `/var/tmp`. The `vi -r` command expects to find preserved files in the `/var/tmp` directory after a system is rebooted.

You can put a size limit on a temporary file system using the `-o size` option of `mount`. See the `tmpfs(7)` manual page for more information. See Chapter 1, “Planning File Systems,” for a description of TMPFS.

▼ How to Create a Temporary File System

To create a temporary file system from a command line:

1. **Become superuser.**
2. **If necessary, create the directory where you want to mount the TMPFS file system and give it the appropriate permissions and ownership.**
3. **Type `mount -F tmp swap directory-name` and press Return.**
The temporary file system is created and mounted on the mount point you specify.

To create an entry for the TMPFS file system in the `/etc/vfstab` file, add an entry like this, separating each field with a Tab:

swap	-	directory	tmp	-	yes	-
------	---	-----------	-----	---	-----	---

Creating a Loopback File System

See Chapter 1, “Planning File Systems,” for a description of loopback file systems.



Caution – Be careful when creating loopback mounts. The potential for confusing both users and applications is enormous. Make sure the loopback entry follows the entries for all other file systems to be included. To be safe, make it the last entry in the `/etc/vfstab` file. If the `/etc/vfstab` entry for the loopback file system precedes the file systems to be included in it, the loopback file system cannot be created.

▼ How to Create a Loopback File System

1. **Become superuser.**
2. **Type `mkdir mount-point` and press Return.**
3. **Type `mount -F lofs lo-directory mount-point`.**
lo-directory specifies the part of the file system to be mounted at the loopback mount point.

If this is to be a permanent loopback file system, you can create an entry in the `/etc/vfstab` file for the loopback file system. Entries for loopback file systems must follow the entries for the file systems to be mounted on the loopback mount point. This example mounts the entire file system hierarchy under the mount point directory `/tmp/newroot`:

#device	device	mount	FS	fsck	auto-	mount
#to mount	to fsck	point	type	pass	mount?	options
/	-	/tmp/newroot	lofs	-	yes	-

This example shows how to use a loopback file system in conjunction with `chroot` to provide a complete virtual file system view to a process or family of processes:

```
# mount -F lofs / /tmp/newroot
# chroot /tmp/newroot command
```


Mounting and Unmounting File Systems

3 

This chapter contains these sections:

<i>Finding Out Which File Systems Are Mounted</i>	<i>page 41</i>
<i>Creating Entries in the File System Table</i>	<i>page 42</i>
<i>Mounting File Systems in the File System Table</i>	<i>page 45</i>
<i>Mounting File Systems From the Command Line</i>	<i>page 47</i>
<i>Unmounting File Systems</i>	<i>page 53</i>
<i>Sharing File Systems</i>	<i>page 55</i>

Finding Out Which File Systems Are Mounted

Sometimes it helps to see which file systems are mounted and to verify that a file system has been mounted or unmounted. Use the `mount` command with no arguments to display a list of mounted file systems. You do not need to be superuser to display a list of mounted file systems.

▼ How to See Which File Systems Are Mounted

◆ **Type `mount` and press Return.**

A list of the file systems currently mounted is displayed.

```

mars% mount
/ on /dev/dsk/c0t3d0s0 read/write on Tue Dec 24 12:29:22 1993
/usr on /dev/dsk/c0t1d0s6 read/write on Tue Dec 24 12:29:22 1993
/proc on /proc read/write on Tue Dec 24 12:29:22 1993
/tmp on swap read/write on Tue Dec 24 12:29:24 1993
/export/home on /dev/dsk/c0t3d0s7 read/write on Tue Dec 24 12:29:22 1993
/usr/openwin on pluto:/export/openwin read/write/remote on Tue Dec 24 12:30:32 1993
/usr/man on pluto:/export/man read/write/remote on Tue Dec 24 12:30:35 1993
/cdrom/ptf_1_2a on /vol/dev/dsk/c1t5/ptf_1_2a read only on Tue Dec 24 12:30:39
mars%

```

Creating Entries in the File System Table

See Chapter 1, “Planning File Systems,” for a description of the `/etc/vfstab` file. This section describes how to create entries in the file system table and provides examples of different types of entries.

▼ How to Create an Entry in the File System Table

1. **Become superuser.**
2. **Edit the `/etc/vfstab` file.**
3. **Add the entry, separating each field with white space (a space or a Tab). If a field has no contents, enter a dash (-).**
4. **Save the changes.**
5. **Check to be sure the mount point directory is present. If not, create it:**
 - a. **Change to the directory where you want to create the mount point.**
 - b. **Type `mkdir directory-name` and press Return.**
6. **Type `mount mount-point` and press Return.**
The entry is mounted.

This example mounts the disk slice `/dev/dsk/c0t3d0s7` as a UFS file system attached to the mount point directory `/files1` with the default mount options (read/write). It specifies the raw character device `/dev/rdisk/c0t3d0s7` as the device to `fsck`. The `fsck pass` value of 2 means that the file system will be checked, but not sequentially.

#device	device	mount	FS	fsck	auto-	mount
#to mount	to fsck	point	type	pass	mount?	options
#						
<code>/dev/dsk/c0t3d0s7</code>	<code>/dev/rdisk/c0t3d0s7</code>	<code>/files1</code>	<code>ufs</code>	<code>2</code>	<code>yes</code>	<code>-</code>

This example mounts the directory `/export/man` from the system `pluto` as an NFS file system on mount point `/usr/man`. You do not specify a device to `fsck` or a `fsck pass` for NFS file systems. In this example, mount options are `ro` (read-only) and `soft`. For greater reliability, specify the hard mount option for read/write NFS file systems.

#device	device	mount	FS	fsck	auto-	mount
#to mount	to fsck	point	type	pass	mount?	options
#						
<code>pluto:/export/man</code>	<code>-</code>	<code>/usr/man</code>	<code>nfs</code>	<code>-</code>	<code>yes</code>	<code>ro,soft</code>

This example mounts the root file system on a loopback mount point named `/tmp/newroot`. Specify `yes` for automount, no device to `fsck`, and no `fsck pass` number. Loopback file systems must always be mounted after the file systems used to make up the loopback file system. Be sure that the loopback entry is the last entry in the `/etc/vfstab` file so that it follows the entries that it depends on.

#device	device	mount	FS	fsck	auto-	mount
#to mount	to fsck	point	type	pass	mount?	options
#						
<code>/</code>	<code>-</code>	<code>/tmp/newroot</code>	<code>lofs</code>	<code>-</code>	<code>yes</code>	<code>-</code>

You do not need to put entries in the `/etc/vfstab` file for CD-ROM or floppy disk devices. CD-ROM devices are mounted automatically by the Volume Management software. Diskettes are mounted after you run the `volcheck(1)` command.

Table 3-1 shows the mount points that volume management uses for CD-ROM and diskettes with file systems. Table 3-2 shows the block and character devices in the `/vol` file system that volume management provides for CD-ROMs and diskettes without file systems.

Table 3-1 CD-ROM and Floppy File System Mount Points

Media Type	Mount Location	State of Media
floppy	<code>/floppy/floppy0</code>	symbolic link to mounted floppy in the first local floppy drive
floppy	<code>/floppy/floppy1</code>	if there are two floppy drives present, symbolic link to mounted floppy in the second local floppy drive
	<code>/floppy/floppy_name</code>	mounted named floppy
	<code>/floppy/unnamed_floppy</code>	mounted unnamed floppy
CD-ROM	<code>/cdrom/cdrom0</code>	symbolic link to mounted CD-ROM in first local CD-ROM drive
CD-ROM	<code>/cdrom/cdrom1</code>	if there is a second CD-ROM drive, symbolic link to mounted CD-ROM in second local CD-ROM drive
	<code>/cdrom/CD-ROM_name</code>	mounted named CD-ROM
	<code>/cdrom/CD-ROM_name/slice</code>	mounted named CD-ROM with a file system
	<code>/cdrom/unnamed_cdrom</code>	mounted unnamed CD-ROM

Table 3-2 CD-ROM and Floppy Device Locations in `/vol`—No File System Present

Media Type	Device Location	State of Media
floppy	<code>/vol/dev/diskette0/unnamed_floppy</code>	formatted unnamed floppy—block device access
	<code>/vol/dev/rdiskette0/unnamed_floppy</code>	formatted unnamed floppy—raw device access
	<code>/vol/dev/diskette0/unlabeled</code>	unlabeled floppy—block device access
	<code>/vol/dev/rdiskette0/unlabeled</code>	unlabeled floppy—raw device access
CD-ROM	<code>/vol/dev/dsk/c0t6/unnamed_cdrom</code>	CD-ROM—block device access
	<code>/vol/dev/rdsk/c0t6/unnamed_cdrom</code>	CD-ROM—raw device access

Mounting File Systems in the File System Table

This section describes how to mount file systems that are included in the `/etc/vfstab` file for a system. The `mountall` command mounts all file systems that have a `yes` in the `automount` field. The `mountall` command is run automatically when entering multiuser run states. Use the `mountall` command to mount all file systems, all local file systems, all remote file systems, or all file systems of a particular file system type in the `/etc/vfstab` file.

Note – The mount-point directory must exist before you can mount any file system on it. To create a mount point, change to the directory where you want to create the mount point, and type `mkdir mount-point` and press Return.

▼ How to Mount File Systems in the File System Table

1. Become superuser.

2. Type `mountall` and press Return.

All the file systems with a `fsck` device entry are checked and fixed, if necessary, before mounting. All file systems listed in the `/etc/vfstab` file with `yes` in the `automount` field are mounted.

```
mars% su
Password:
# mountall
#
```

If file systems are already mounted, messages are displayed telling you so.

```
mars% su
Password:
# mountall
mount: /tmp already mounted
nfs mount: mount: /usr/openwin: Device busy
nfs mount: mount: /usr/man: Device busy
#
```

▼ How to Mount Local File Systems in the File System Table

1. Become superuser.

2. Type `mountall -l` and press Return.

All the local file systems with a `fsck` device entry are checked and fixed, if necessary, before mounting. All local file systems listed in the `/etc/vfstab` file with `yes` in the `automount` field are mounted.

```
# mountall -l
# mount
/ on /dev/dsk/c0t3d0s0 read/write on Tue Dec 24 12:29:22 1993
/usr on /dev/dsk/c0t1d0s6 read/write on Tue Dec 24 12:29:22 1993
/proc on /proc read/write on Tue Dec 24 12:29:22 1993
/tmp on swap read/write on Mon Dec 30 12:37:33 1993
#
```

▼ How to Mount Remote File Systems in the File System Table

1. Become superuser.

2. Type `mountall -r` and press Return.

All remote file systems listed in the `/etc/vfstab` file with `yes` in the `automount` field are mounted.

```
# mountall -r
# mount
/ on /dev/dsk/c0t3d0s0 read/write on Tue Dec 24 12:29:22 1993
/usr on /dev/dsk/c0t1d0s6 read/write on Tue Dec 24 12:29:22 1993
/proc on /proc read/write on Tue Dec 24 12:29:22 1993
/tmp_mnt/home/neptune on neptune:/export/home/neptune read/write/remote on Mon Dec 30
12:27:41 1993
/usr/openwin on pluto:/export/openwin read/write/remote on Mon Dec 30 12:37:53 1993
/usr/man on pluto:/export/man read/write/remote on Mon Dec 30 12:37:55 1993
#
```


▼ How to Mount File Systems in the File System Table by File System Type

1. Become superuser.

2. Type `mountall -F fstype` and press Return.

All the file systems of the type you specify with a `fsck` device entry are checked and fixed, if necessary, before mounting. All file systems of the type you specify listed in the `/etc/vfstab` file with `yes` in the `automnt` field are mounted.

The following example mounts all the NFS file systems listed in the `/etc/vfstab` file:

```
# mountall -F nfs
# mount
/ on /dev/dsk/c0t3d0s0 read/write on Tue Dec 24 12:29:22 1993
/usr on /dev/dsk/c0t1d0s6 read/write on Tue Dec 24 12:29:22 1993
/proc on /proc read/write on Tue Dec 24 12:29:22 1993
/tmp_mnt/home/neptune on neptune:/export/home/neptune read/write/remote on Mon Dec 30
12:27:41 1993
/usr/openwin on pluto:/export/openwin read/write/remote on Mon Dec 30 12:49:09 1993
/usr/man on pluto:/export/man read/write/remote on Mon Dec 30 12:49:11 1993
#
```

Mounting File Systems From the Command Line

This section describes how to mount file systems from a command line.

Note – The mount-point directory must exist before you can mount any file system on it. To create a mount point, change to the directory where you want to create the mount point, and type `mkdir mount-point` and press Return.

▼ How to Mount a UFS File System With the Default Options

To mount a UFS file system with the default options, specify the block device name of the slice from the `/dev/dsk` directory and the mount point for the file system. `device-name` specifies the special block device file for the disk slice

holding the file system (for example, `/dev/dsk/c0t3d0s7`). See Chapter 1, “Planning File Systems,” for information about how to find out disk device names. *mount-point* specifies where the file system is mounted.

1. Become superuser.

2. Type `mount /dev/dsk/device-name mount-point` and press Return.
The file system is mounted.

In this example, `/dev/dsk/c0t3d0s7` is mounted on the `/files1` directory:

```
pluto& su
Password:
# mount /dev/dsk/c0t3d0s7 /files1
#
```

▼ How to Mount a File System That Has an `/etc/vfstab` Entry

If there is an entry for the file system in `/etc/vfstab`, you can specify either the mount point or the block device. It is usually easier to specify the mount point. The rest of the information is obtained from `/etc/vfstab`.

1. Become superuser.

2. Type `mount mount-point` and press Return.
The file system is mounted.

In this example, `/usr/openwin` is mounted:

```
pluto& su
Password:
# mount /usr/openwin
#
```

▼ How to Use Options to Mount a UFS File System

You can mount a UFS file system with the mount options shown in Table 3-3. If you specify multiple options, separate them with commas (no spaces). For example, `-o ro,nosuid`.

Table 3-3 UFS-Specific Mount Options

Option	Description
<code>f</code>	Fake an entry in <code>/etc/mnttab</code> , but don't really mount any file systems
<code>n</code>	Mount the file system without making an entry in <code>/etc/mnttab</code>
<code>rw</code>	Read/write
<code>ro</code>	Read-only. If you do not specify this option, the default is read/write.
<code>nosuid</code>	Disallow <code>setuid</code> execution and prevent devices on the file system from being opened. The default is to allow <code>setuid</code> execution and allow devices to be opened.
<code>remount</code>	Used together with <code>rw</code> to remount a file system read/write

See the `mount_ufs(1M)` manual page for a complete list of options.

◆ **Type** `mount -o options /dev/dsk/device-name mount-point`.
The file system is mounted using the options you specify.

A CD-ROM that contains a file system is automatically mounted by the volume management software. A diskette containing a file system is mounted when you run the `volcheck(1)` command.

▼ How to Mount an NFS File System

To mount an NFS file system, it must be made available from the server system. The `share(1M)` command creates a list of file systems in the file `/etc/dfs/sharetab` than can be shared across the network. See “Sharing File Systems” on page 55 for information on how to export file systems.

To mount an available NFS file system:

- Become superuser.**
- Type** `mount host:/directory mount-point` and press **Return**.
The file system is mounted.

In this example, manual pages from the host system `pluto` in the directory `/export/man` are mounted on `/usr/man`:

```
pluto& su
Password:
# mount pluto:/export/man /usr/man
#
```

Note – This example is probably not very realistic. Manual pages can be more effectively accessed by including them in the `/etc/vfstab` file or by making them available through `autofs`.

▼ How to Use Options to Mount an NFS File System

You can mount an NFS file system with the mount options shown in Table 3-4. If you specify multiple options, separate them with commas (no spaces). For example :

```
-o ro,nosuid,retry=500
```

Table 3-4 NFS-Specific Mount Options

Option	Description
<code>rw ro</code>	Read/write or read-only. If you do not specify this option, the default is read/write.
<code>nosuid</code>	Disallow <code>setuid</code> execution. The default is to allow <code>setuid</code> execution.
<code>remount</code>	If a file system is mounted read-only, remounts the file system read/write.

Table 3-4 NFS-Specific Mount Options (Continued)

Option	Description
<code>bg</code> <code>fg</code>	If the first attempt fails, retry in the background or in the foreground. The default is <code>fg</code> .
<code>soft</code> <code>hard</code>	Soft indicates that an error is returned if the server does not respond. Hard indicates that the retry request is continued until the server responds. The default is <code>hard</code> .
<code>intr</code> <code>nointr</code>	Allow do not allow keyboard interrupts to kill a process that is hung while waiting for a response on hard-mounted file systems. The default is <code>intr</code> .

See the `mount_nfs(1M)` manual page for a complete list of options.

1. Become superuser.

2. Type `mount -F nfs -o options host: /directory mount-point` and press Return.

The file system is mounted using the options you specify.

▼ How to Mount HSFS File Systems

CD-ROM devices that contain file systems are mounted automatically by the Volume Management software. Table 3-5 shows the mount points for named and unnamed CD-ROM devices.

Table 3-5 CD-ROM Mount Points

Mount Point	State of the Media
<code>/cdrom/cdrom0</code>	Symbolic link to mounted CD-ROM in first local CD-ROM drive
<code>/cdrom/CD_ROM_name</code>	Mounted named CD-ROM
<code>/cdrom/CD_ROM_name/slice</code>	Mounted named CD-ROM with a file system
<code>/cdrom/unnamed_cdrom</code>	Mounted unnamed CD-ROM

Some CD-ROMs contain a mixture of HSFS and UFS slices. The volume management software mounts all HSFS and UFS file systems.

▼ How to Mount PCFS File Systems

To mount a diskette containing a file system, you must run the `volcheck(1)` command. Table 3-6 shows the mount points for named and unnamed diskettes.

Table 3-6 Diskette Mount Points

Mount Point	State of the Media
<code>/floppy/floppy0</code>	Symbolic link to mounted diskette in first local floppy drive
<code>/floppy/floppy_name</code>	Mounted named diskette
<code>/floppy/unnamed_floppy</code>	Mounted unnamed diskette

See the *Administration Supplement for Solaris Platforms* for information on mounting a PCFS on a hard disk.

▼ How to Mount an `s5fs` File System With Default Options

To mount an `s5fs` file system with the default options, specify the block device name of the partition from the `/dev/dsk` directory and the mount point for the file system. *device-name* specifies the special block device file for the disk partition holding the file system (for example, `/dev/dsk/c0t3d0s7`). *mount-point* specifies where the file system is mounted.

1. **Become superuser.**
2. **Type `mount -F s5fs /dev/dsk/device-name mount-point` and press Return.**
The file system is mounted.

In this example, `/dev/dsk/c0t3d0s7` is mounted on the `/files1` directory:

```
pluto% su
Password:
# mount -F s5fs /dev/dsk/c0t3d0s7 /files1
```

Unmounting File Systems

You cannot unmount a file system that is busy, that is, when a user has changed directory into the file system or a file in the file system is open.

Notify users if you need to unmount a file system they are using. Suggested ways to make a file system available for unmounting are:

- Change to a directory in a different file system
- Log out of the system
- Use the `fuser` command to list all processes accessing the file system and to kill them if necessary

The output of the `fuser` command is a series of numbers, each with a one-letter code that identifies the way that the process is using the file. Table 3-7 describes these codes.

Table 3-7 Codes Used by the `fuser` Command

Code	
c	Current directory
r	Root directory
o	Open file

See the `fuser(1M)` manual page for more information.

▼ How to Terminate All Processes for a File System

Note – You should not kill a user's processes without warning.

1. **Become superuser.**
2. **Type `fuser -c mount-point` and press Return.**
A list of processes is displayed.
3. **Type `fuser -k mount-point` and press Return.**
A SIGKILL is sent to each process.

▼ How to Unmount File Systems

To unmount a file system (except / or /usr):

Note – The root (/) and /usr UFS file systems are special cases. The root file system can be unmounted only during a shutdown, since the system needs root to function.

1. Become superuser.

2. Type `cd directory` and press Return.

The directory must be a file system other than the one to be unmounted.

3. Type `umount mount-point` and press Return.

The file system is unmounted, or an error message is displayed.

For example, to unmount a local home directory:

```
pluto% su
Password:
# cd /
# umount /home
#
```

Alternatively, you can specify the block device for UFS, PCFS, or HSFS file systems, the resource for an NFS file system, or the loopback directory for LOFS file systems.

This example unmounts the file system in slice 7:

```
pluto% su
Password:
# cd /
# umount /dev/dsk/c0t0d0s7
#
```


▼ How to Unmount File Systems Listed in `vfstab`

To unmount all the file systems listed in the `vfstab`, except `/`, `/proc`, `/var`, and `/usr`:

1. **Become superuser.**
2. **Type `umountall` and press Return.**
All systems that can be unmounted are unmounted. File systems that are busy are not unmounted.
3. **Communicate with users that you need to unmount their file systems, and repeat Step 2.**
4. **If necessary, type `killall 9` and press Return.**
All active processes except those needed to shut down the system are killed.
5. **Repeat Step 2 as needed until all file systems are unmounted.**

To return to multiuser mode:

- ♦ **Type `init 3` and press Return.**

Sharing File Systems

This chapter describes how to set up automatic sharing of file systems and provides some examples.

▼ How to Set Up Automatic Sharing

1. **Become superuser.**
2. **Edit the `/etc/dfs/dfstab` file.**
Add one entry to the file for each resource that you wish to have shared automatically. Each entry must be on a line by itself in the file and use the following syntax:

```
share [-F fs-type] [-o specific-options] [-d description] pathname
```

Examples of Automatic Sharing Entries in the /etc/dfs/dfstab File

If you wanted to permit the root user on `samba` to always have root access to the `/usr/src` on the server machine, you would make the following entry to the server `dfstab` file:

```
share -F nfs -o root=samba /usr/src
```

If you wanted to permit the root users on `samba`, `homedog`, and `chester` to always have root access to the `/usr/src` directory on the server machine, you would make the following entry to the server `dfstab` file:

```
share -F nfs -o root=samba:homedog:chester /usr/src
```

If you wanted all client processes with UID 0 to have superuser access to `/usr/src`, you would make the following entry in the server `dfstab` file:

```
share -F nfs -o anon=0 /usr/src
```

`anon` is short for “anonymous.” Anonymous requests, by default, get their user ID changed from its previous value (whatever it may be) to the user ID of user name `nobody`. NFS servers label as anonymous any request from a root user (someone whose current effective user ID is 0) who is not in the list following the `root=option` in the `share` command. The command above tells the kernel to use the value 0 for anonymous requests. The result is that all root users retain their user ID of 0.

Note – NFS is the most common distributed file system type, as illustrated in these examples.

Copying UFS Files and File Systems



This chapter contains these sections:

<i>Copying Complete File Systems</i>	<i>page 58</i>
<i>Copying Files and File Systems to Tape</i>	<i>page 65</i>
<i>Copying Files and File Systems to Diskette</i>	<i>page 72</i>
<i>Copying Files with a Different Header Format</i>	<i>page 76</i>
<i>Retrieving Files Created With the bar Command</i>	<i>page 77</i>

Commands for Copying File Systems

When you need to back up and restore complete file systems, use the `ufsdump` and `ufsrestore` commands described in Part 2 — “Backup and Restore.”

When you want to copy or move individual files, portions of file systems, or complete file systems, you can use the procedures described in this chapter as an alternative to `ufsdump` and `ufsrestore`. Table 4-1 shows the tasks described in this chapter and the commands that you can use for each task.

Table 4-1 Summary of Tasks and Available Commands

Task	Commands
Copying complete file systems	<code>labelit</code> <code>volcopy</code> <code>dd</code> <code>cpio</code>
Copying files or file systems to tape or diskette	<code>tar</code> <code>cpio</code>
Copying files between systems running different SunOS release levels	<code>tar</code> <code>cpio</code>
Copying file archives with different header formats	<code>cpio -H <i>format</i></code>
Copying files to a group of systems	<code>rdist</code>

Copying Complete File Systems

You can copy complete file systems in three ways:

- “Using the `labelit` and `volcopy` Commands” on page 59
- “Using the `dd` Command” on page 63
- “Using the `cpio` Command” on page 64

These sections describe the advantages and disadvantages of each method and provide examples of how to use the commands.

Using the `labelit` and `volcopy` Commands

The `volcopy` command makes a literal (block) copy or image of a complete UFS file system on another file system or to a tape. Use this command to make a fast copy of a file system. This kind of image is a literal “snapshot” of the file system.

Note – Making a `volcopy` image can take quite a bit of time (sometimes hours) because the `volcopy` command does not write efficiently to streaming tape cartridges.

You must restore the complete `volcopy` image; you cannot restore only a part of the image. The `volcopy` command checks the file system label you create using the `labelit(1M)` command (explained in “Assigning Labels to UFS File Systems” on page 59) and guarantees that the correct volume is mounted. Use the `volcopy` command to extract a volume that was created with the `volcopy` command.

Note – `volcopy` cannot be used to copy file systems from one tape to another.

Assigning Labels to UFS File Systems

You can use the `labelit` command to write a file system name and a volume name of up to six characters to the superblock of each UFS file system. These file system and volume names can be used as an alternative to specifying the raw character device name as an argument to the `volcopy` command.

Note – The file system volume name written to the superblock by the `labelit` command is completely separate from the slice label in the VTOC (volume table of contents) of a file system.

The volume label can be very useful in matching volumes stored on disk packs, diskettes, or tape.

▼ **How to Find the File System and Volume Name**

1. Become superuser.

2. Type `labelit /dev/rdisk/cntndnsn` and press Return.

The file system name and the volume name are displayed. If no volume name is assigned, the fields are blank.

In this example, no volume name is assigned:

```
mars% su
Password:
# labelit /dev/rdisk/c0t3d0s7
fsname:
volume:
#
```

▼ **How to Assign a File System and Volume Name**

1. Become superuser.

2. Type `labelit -F ufs /dev/rdisk/cntndnsn fs-name volume-name` and press Return.

The file system name and the volume name of up to six characters are assigned and displayed.

In this example, the file system name `home1` and the volume name `voll` are assigned:

```
mars% su
Password:
# labelit -F ufs /dev/rdisk/c0t030s7 home1 voll
fsname: home1
volume: voll
#
```

See the `labelit(1M)` manual page for more information.

▼ How to Copy File Systems With `volcopy`

Note – It is just as important for file systems to be quiescent when using the `volcopy` command as it is when doing backups. See Chapter 3, “Mounting and Unmounting File Systems,” for information about unmounting file systems.

1. **Become superuser.**
2. **Unmount the file system.**
3. **Type `volcopy -F ufs fs-name source-device volume-name1 destination-device volume-name2` and press Return.**
4. **If you have assigned new labels or changed old ones, you will be asked to confirm that the new information is correct.**
5. **Press Return to confirm that you want to copy the volume from the slice to the device you specified.**

Example: How to Copy File Systems With `volcopy`

First, use `labelit` to create a file system name and volume name for the file system:

```
# labelit -F ufs /dev/dsk/c0t3d0s7 home1 vol1
fsname: home1
volume: vol1
#
```

Next, copy an image of the file system to a tape device, giving it a new volume name, `tape1`:

Code Example 4-1 Copying File Systems with `volcopy`

```
mars% su
Password:
mars# cd /
mars# umount /dev/dsk/c0t3d0s7
mars# volcopy -F ufs home1 /dev/rdisk/c0t3d0s7 vol1 \ /dev/rmt/0
tape1
```

Code Example 4-1 Copying File Systems with `volcopy` (Continued)

```
arg. (voll) doesn't agree with from vol.(voll)
Type 'y' to override:  y
/dev/rmt/0 less than 48 hours older than /dev/rdisk/c0t3d0s7
To filesystem dated:  Thu Mar 12 14:34:48 1992
Type 'y' to override:  y
arg.(tapel) doesn't agree with to vol.(      )
Type 'y' to override:  y
warning! from fs(homel) differs from to fs(      )
Type 'y' to override:  y
From: /dev/rdsk/c0t3d0s7, to: /dev/rmt/0? (DEL if wrong)
mars#
```

This example shows how `volcopy` performs a check between the source device and the destination device using the file system name and the volume name. The source volume has no existing label, so you specify its volume and file system name as empty strings. The tape already contained a `volcopy` image.

```
# volcopy -F ufs "" /dev/rdsk/c0t3d0s0 "" /dev/rmt/01 tapel
/dev/rmt/01 less than 48 hours older than /dev/rdsk/c0t3d0s0
To filesystem dated:  Sat Jan 4 14:42:25 1992
Type 'y' to override:  y
arg.(tapel) doesn't agree with to vol.(  )
Type 'y' to override:  y
warning! from fs() differs from to fs(  )
Type 'y' to override:  y
From: /dev/rdsk/c0t3d0s0, to: /dev/rmt/01? (DEL if wrong)  y
```

See the `volcopy_ufs(1M)` manual page for more information.

Using the `dd` Command

The `dd` command makes a literal (block) copy of a complete UFS file system to another file system or to a tape. By default, the `dd` command copies its standard input to its standard output, so you can copy a file with this command:

```
dd < in-file > out-file
```

Note – The `dd` command is not recommended for use with variable-length tape drives.

You can specify a device name in place of the standard input or the standard output or both. In this example, contents of the diskette are copied to a file in the `/tmp` directory:

```
$ dd < /floppy/floppy0 > /tmp/output.file
2400+0 records in
2400+0 records out
$
```

The `dd` command reports on the number of blocks it reads and writes. The number after the `+` is a count of the partial blocks that were copied.

The `dd` command syntax is different from most other commands. Options are specified as *keyword=value* pairs, where *keyword* is the option you want to set and *value* is the argument for that option. For example, you can replace the standard input and output with this syntax:

```
dd if=input-file of=output-file
```

For example, to use the *keyword=value* pairs instead of the redirect symbols in the previous example, you would type:

```
$ dd if=/floppy/floppy0 of=/tmp/output.file
```

See the `dd(1M)` manual page for more information.

Using the `cpio` Command

You can use the `cpio` (copy in and out) command to copy individual files, groups of files, or complete file systems. This section describes how to use the `cpio` command to copy complete file systems.

The `cpio` command is an archiving program that takes a list of files and copies them into a single, large output file. It inserts headers between the individual files to facilitate recovery. You can use the `cpio` command to copy complete file systems to another slice, another system, or to a media device such as tape or diskette.

Because the `cpio` command recognizes end-of-media and prompts you to insert another volume, it is the most effective command (other than `ufsdump`) to use to create archives that require multiple tapes or diskettes.

You frequently use commands like `ls` and `find` to list and select the files you want to copy and then pipe the output to the `cpio` command.

▼ How to Copy Directory Trees Between File Systems

1. Become superuser.

2. Type `cd /filesystem1` and press Return.

3. Type `find . -print -depth | cpio -pdm /filesystem2` and press Return.

The files from the directory name you specify are copied and symbolic links are preserved. The `.` argument to `find` copies the current working directory, the `-print` option prints the file names, and the `-depth` option descends the directory hierarchy and prints file names on the way back up, which allows the `-m` option to `cpio` to set the correct modification times on directories. The `-p` option to `cpio` creates a list of files. The `d` option creates directories as needed.

4. To verify that the copy was successful, type `cd /filesystem2;ls` and press Return.

5. If appropriate, to remove the source directory tree, type `rm -rf /filesystem1` and press Return.

See the `cpio(1)` manual page for more information.

Copying Files and File Systems to Tape

`tar` and `cpio` are commands that can be used to copy files and file systems to tape. The command you choose depends on how much flexibility and precision you require for the copy.

Use `tar` to copy files and directory subtrees to a single tape. Note that the SunOS 5.x `tar` command can archive special files (block and character devices, `fifos`) but the SunOS 4.x `tar` command cannot extract them. The `cpio` command provides better portability.

Use `cpio` to copy arbitrary sets of files, special files, or file systems that require multiple tape volumes or when you want to copy files from SunOS 5.x systems to SunOS 4.x systems. The `cpio` command packs data onto tape more efficiently than `tar` and skips over any bad spots in a tape when restoring. The `cpio` command also provides options for writing files with different header formats (`tar`, `ustar`, `crc`, `odc`, `bar`) for portability between systems of different types.

Because `tar` and `cpio` use the raw device, you do not need to format or make a file system on tapes before you use them. The tape drive and device name you use depend on the hardware and configuration for each system. See “Choosing Which Media to Use” on page 102 for more information about tape drives and device names.

The information is divided into these sections:

- “Useful Commands for Streaming Tape Cartridges” on page 65
- “Using the `tar` Command” on page 67
- “Using the `cpio` Command” on page 69

Useful Commands for Streaming Tape Cartridges

This section contains a few commands for use with streaming tape cartridges.

▼ How to Retension a Magnetic Tape Cartridge

If errors occur when reading a tape, retension the tape, clean the tape drive, and then try again.

◆ **Type** `mt -f /dev/rmt/n retension` **and press Return.**

The tape in the specified tape drive is retensioned.

In this example, the tape in drive `/dev/rmt/1` is retensioned:

```
venus% mt -f /dev/rmt/1 retension
venus%
```

Note – The retension option is not recommended for non-QIC tape drives.

▼ **How to Rewind a Magnetic Tape Cartridge**

- ◆ **Type** `mt -f /dev/rmt/n rewind` **and press Return.**
The tape in the specified tape drive is rewound.

In this example, the tape in drive `/dev/rmt/1` is rewound:

```
venus% mt -f /dev/rmt/1 rewind
venus%
```

▼ **How to Show the Status of a Magnetic Tape Drive**

- ◆ **Type** `mt -f /dev/rmt/n status` **and press Return.**
Status for the tape drive you specify is displayed.

In this example, there is no tape in drive `/dev/rmt/1`:

```
venus% mt -f /dev/rmt/1 status
/dev/rmt/1: no tape loaded or drive offline
venus%
```

In this example, status is shown for the tape in drive `/dev/rmt/1`:

```
venus% mt -f /dev/rmt/1 status
Archive QIC-150 tape drive:
  sense key(0x6)= unit attention   residual= 0   retries= 0
  file no= 0   block no= 0
venus%
```

Using the `tar` Command

▼ How to Copy Files to a Tape (`tar`)

1. Change to the directory that contains the files you want to copy.
2. Insert a write-enabled tape into the tape drive.



Caution – Copying files to a tape using the `c` option to `tar` destroys any files already on the tape. If you want to preserve the files already on the tape, use the `r` option described in “How to Append Files to a Tape (`tar`)” on page 68.

3. Type `tar cvf /dev/rmt/n filename filename filename ...` and press **Return**. The `c` option indicates that you want to create an archive. The `v` option displays the name of each file as it is archived. The `f` option indicates that the archive should be written to the specified device or file. The file names you specify are copied to the tape, overwriting any existing files on the tape.

Note – You can use metacharacters (`?` and `*`) as part of the file names you specify. For example, to copy all documents with a `.doc` suffix, type `*.doc` as the file name argument.

4. Remove the tape from the drive and write the names of the files on the tape label.

In this example, two files are copied to a tape in tape drive 0:

```
venus% cd /home/smith
venus% ls evaluation*
evaluation.doc    evaluation.doc.backup
venus% tar cvf /dev/rmt/0 evaluation*
a evaluation.doc 86 blocks
a evaluation.doc.backup 84 blocks
venus%
```

▼ How to List the Files on a Tape (`tar`)

1. Insert a tape into the tape drive.

2. Type `tar tvf /dev/rmt/n` and press Return.

The `t` option lists the table of contents for the files on the tape in the tape drive you specify.

In this example, the table of contents for the tape in drive 0 contains two files:

```
venus% tar tvf /dev/rmt/0
rw-rw-rw-6693/10  44032 Apr 23 14:54 1994 evaluation.doc
rw-rw-rw-6693/10  43008 Apr 23 14:47 1994 evaluation.doc.backup
venus%
```

▼ How to Append Files to a Tape (`tar`)**1. Change to the directory that contains the files you want to copy.****2. Insert a tape that is not write-protected into the tape drive.****3. Type `tar rvf /dev/rmt/n filename filename filename ...` and press Return.**

The `r` option indicates that the files should be appended to an existing archive. The specified files are appended to the files already on the tape.

Note – You can use metacharacters (`?` and `*`) as part of the file names you specify. For example, to copy all documents with a `.doc` suffix, type `*.doc` as the file name argument.

4. Remove the tape from the drive and write the names of the files on the tape label.

In this example, one file is appended to the files already on the tape in drive 0:

```
venus% cd /home/smith
venus% tar rvf /dev/rmt/0 junk
a junk 1 blocks
venus% tar tvf /dev/rmt/0
rw-rw-rw-6693/10  44032 Apr 23 14:54 1994 evaluation.doc
rw-rw-rw-6693/10  43008 Apr 23 14:47 1994 evaluation.doc.backup
rw-rw-rw-6693/10      18 Dec 10 11:36 1993 junk
venus%
```

▼ **How to Retrieve Files From a Tape (tar)**

1. Change to the directory where you want to put the files.

2. Insert the tape into the tape drive.

3. Type `tar xvf /dev/rmt/n` and press Return.

The `x` option indicates that files should be extracted from the specified archive file. All of the files on the tape in the specified drive are copied to the current directory.

In this example, all files are copied from the tape in drive 0:

```
venus% cd /home/smith/Evaluations
venus% tar xvf /dev/rmt/0 evaluation*
x evaluation.doc, 44032 bytes, 86 tape blocks
x evaluation.doc.backup, 43008 bytes, 84 tape blocks
venus%
```

To retrieve individual files from a tape:

◆ **Type `tar xvf /dev/rmt/n filename filename filename ...` and press Return.**

The file names you specify are extracted from the tape and placed in the current working directory.

Note – The names of the files extracted from the tape exactly match the names of the files stored on the archive. If you have any doubts about the names or paths of the files, first list the files on the tape. See “How to List the Files on a Tape (tar)” on page 67 for instructions.

See the `tar(1)` manual page for more information.

Using the `cpio` Command

The `cpio` command, used to create an archive, takes a list of files or path names from standard input and writes to standard output. The output is almost always redirected to a file or to a device.

▼ How to Copy All Files in a Directory to a Tape (cpio)

- 1. Insert a tape that is not write-protected into the tape drive.**
- 2. Type `ls | cpio -oc > /dev/rmt/n` and press Return.**
All of the files in the directory are copied to the tape in the drive you specify, overwriting any existing files on the tape. The total number of blocks copied is displayed.
- 3. Remove the tape from the drive and write the names of the files on the tape label.**

In this example, all of the files in the directory `/home/smith/TOI` are copied to the tape in tape drive 0:

```
venus% cd /home/smith/TOI
venus% ls | cpio -oc > /dev/rmt/0
31 blocks
venus%
```

▼ How to List the Files on a Tape (cpio)

Note – Listing the table of contents takes as long as it does to read the archive file because the `cpio` command must process the entire archive.

- 1. Insert a tape into the tape drive.**
- 2. Type `cpio -civt < /dev/rmt/n` and press Return.**
The `i` option reads in the contents of the tape. The `v` option displays the output in a format similar to the output from the `ls -l` command. The `t` option lists the table of contents for the files on the tape in the tape drive you specify.

In this example, the table of contents for the tape in drive 0 contains four files:

```
venus% cpio -civt < /dev/rmt/0
100666 smith  3895  Feb 24 15:13:02 1992  Boot.chapter
100666 smith  3895  Feb 24 15:13:23 1992  Directory.chapter
100666 smith  6491  Feb 24 15:13:52 1992  Install.chapter
```



```
100666 smith  1299  Feb 24 15:14:00 1992  Intro.chapter
31 blocks
venus%
```

▼ How to Retrieve All Files From a Tape (cpio)

If the archive was created using relative path names, the input files are built as a directory within the current directory. If, however, the archive was created with absolute path names, the same absolute paths are used to re-create the file.



Caution - Using absolute path names can be dangerous because you will overwrite the original files.

1. Change to the directory where you want to put the files.

2. Insert the tape into the tape drive.

3. Type `cpio -icv < /dev/rmt/n` and press Return.

All of the files on the tape in the drive you specify are copied to the current directory.

In this example, all files are copied from the tape in drive 0:

```
venus% cpio -icv < /dev/rmt/0
Boot.chapter
Directory.chapter
Install.chapter
Intro.chapter
31 blocks
venus%
```

▼ How to Retrieve a Subset of Files From a Tape (cpio)

You can reload a subset of the files from the archive by specifying a pattern to match using shell wild card characters enclosed in quotes after the options.

1. Change to the directory where you want to put the files.

2. Insert the tape into the tape drive.

3. Type `cpio -icv "*file" < /dev/rmt/n` and press Return.

All of the files that match the pattern are copied to the current directory. You can specify multiple patterns, but each must be enclosed in double quotation marks.

In this example, all files that end in the suffix `chapter` are copied from the tape in drive 0:

```
venus% cd /home/smith/Book
venus% cpio -icv "*chapter" < /dev/rmt/0
Boot.chapter
Directory.chapter
Install.chapter
Intro.chapter
31 blocks
venus%
```

See the `cpio(1)` manual page for more information.

Copying Files and File Systems to Diskette

Before you can copy files or file systems to diskette, you must format the diskette. See the *Administration Supplement for Solaris Platforms* for information on how to format a diskette. Use the `tar` command to copy UFS files to a single formatted diskette. Use the `cpio` command if you need to copy UFS files to multiple formatted diskettes. `cpio` recognizes end-of-media and prompts you to insert the next volume.

Use double-sided high-density 3.5-inch diskettes (diskettes are marked "DS, HD").

How to Get a Diskette Out of the Drive

See the *Administration Supplement for Solaris Platforms* for information on getting a diskette out of the drive.

▼ How to Copy Files to a Single Formatted Diskette

Note – If the diskette contains a mounted file system, you must unmount the filesystem before running `tar`. You can unmount the file system using the `umount` command, or you can unmount the file system and format the diskette using the `fdformat -U` command.

1. **Change to the directory that contains the files you want to copy.**
2. **Insert a formatted diskette that is not write-protected into the drive.**



Caution – Copying files to a formatted diskette using the `c` option of `tar` destroys any files already on the diskette. If you want to preserve the files already on the diskette, use the `tar r` option described in “How to Append Files to a Formatted Diskette” on page 74.

3. **Type `volcheck` and press Return.**
The `volcheck` command makes the diskette available.
4. **If the diskette contains a PCFS, type `fdformat -U /vol/dev/aliases/floppy0` and press Return.**
The diskette is reformatted and unmounted.



Caution – Reformatting a diskette destroys any files or archives that were already on the diskette.

5. **Type `tar cvf /vol/dev/rdiskette0/unlabeled filename filename filename ...` and press Return.**
The file names you specify are copied to the diskette, overwriting any existing files on the diskette.
6. **Remove the diskette from the drive.**
7. **Write the names of the files on the diskette label.**

In this example, two files are copied to a diskette:

```
venus% cd /home/smith
venus% ls evaluation*
evaluation.doc  evaluation.doc.backup
venus% tar cvf /vol/dev/rdiskette0/unlabeled evaluation*
a evaluation.doc 86 blocks
a evaluation.doc.backup 84 blocks
venus%
```

▼ How to List the Files on a Diskette

1. **Insert a diskette into the drive.**
2. **Run `volcheck(1)` to make the diskette available.**
3. **Type `tar tvf /vol/dev/rdiskette0/unlabeled` and press Return.**
The `t` option lists the table of contents for the files on the diskette.

In this example, the table of contents for the diskette contains two files:

```
venus% tar tvf /vol/dev/rdiskette0/unlabeled
rw-rw-rw-6693/10  44032 Apr 23 14:54 1994 evaluation.doc
rw-rw-rw-6693/10  43008 Apr 23 14:47 1994 evaluation.doc.backup
venus%
```

See the `tar(1)` manual page for more information.

If you need a multiple-volume interchange utility, use `cpio`. `tar` is only a single-volume utility.

▼ How to Append Files to a Formatted Diskette

1. **Change to the directory that contains the file you want to copy.**
2. **Insert a formatted diskette that is not write-protected into the drive.**
3. **Type `tar rvf /vol/dev/rdiskette0/unlabeled filename filename filename ...` and press Return.**
The file names you specify are appended to the files already on the diskette.

Note – You can use metacharacters (? and *) as part of the file names you specify. For example, to copy all documents with a .doc suffix, type *.doc as the file name argument.

4. Remove the diskette from the drive.

5. Write the names of the files on the diskette label.

In this example, one file is appended to the files already on the diskette:

```
venus% cd /home/smith
venus% tar rvf /vol/dev/rdiskette0/unlabeled junk
a junk 1 blocks
venus% tar tvf /vol/dev/rdiskette0/unlabeled
rw-rw-rw-6693/10  44032 Apr 23 14:54 1994 evaluation.doc
rw-rw-rw-6693/10  43008 Apr 23 14:47 1994 evaluation.doc.backup
rw-rw-rw-6693/10    18 Dec 10 11:36 1993 junk
venus%
```

▼ **How to Retrieve Files From a Diskette**

1. Change to the directory where you want to put the files.

2. Insert the diskette into the drive.

3. Run volcheck(1) to make the diskette available.

4. Type tar xvf /vol/dev/rdiskette0/unlabeled and press Return.
All of the files on the diskette are copied to the current directory.

5. Remove the diskette from the drive.

In this example, all files are copied from the diskette:

```
venus% cd /home/smith/Evaluations
venus% tar xvf /vol/dev/rdiskette0/unlabeled evaluation*
x evaluation.doc, 44032 bytes, 86 tape blocks
x evaluation.doc.backup, 43008 bytes, 84 tape blocks
venus%
```

To retrieve individual files from a diskette:

♦ **Type** `tar xvf /vol/dev/rdiskette0/unlabeled filename filename filename ...` **and press Return.**

The file names you specify are extracted from the diskette and placed in the current working directory.

▼ How to Archive Files to Multiple Diskettes

If you are copying large files or file systems onto diskettes, you want to be prompted to replace a full diskette with another formatted diskette. The `cpio` command provides this capability. The `cpio` commands you use are the same as you would use to copy files to tape, except you would specify `/vol/dev/aliases/floppy0` as the device instead of the tape device name. See “Using the `cpio` Command” on page 69 for information on how to use `cpio`.

Copying Files with a Different Header Format

Archives created with the SunOS 5.x `cpio` command may not be compatible with older SunOS releases. The `cpio` command allows you to create archives that can be read with several other formats. You specify these formats using the `-H` option and one of these arguments:

- `crc` or `CRC` – ASCII header with checksum
- `ustar` or `USTAR` – IEEE/P1003 Data Interchange
- `tar` or `TAR` – tar header and format
- `odc` – ASCII header with small device numbers
- `bar` – bar header and format

The syntax for using the header options is:

```
cpio -o -H header-option < file-list > output-archive
```

▼ How to Create an Archive Compatible with Older SunOS Releases

♦ **Type** `cpio -oH odc < file-list > /dev/rmt/n` and press **Return**.

The `-H` options have the same meaning for input as they do for output. If the archive was created using the `-H` option, you must use the same option when the archive is read back in or the `cpio` command will fail, as shown in this example:

```
venus% find . -print | cpio -oH tar > /tmp/test
113 blocks
venus% cpio -iH bar < /tmp/test
cpio: Invalid header "bar" specified
USAGE:
    cpio -i[bcdfkmrstuvBSV6] [-C size] [-E file] [-H hdr] [-I
file [-M msg]] [-R id] [patterns]
    cpio -o[acvABLV] [-C size] [-H hdr] [-O file [-M msg]]
    cpio -p[adlmuvLV] [-R id] directory
venus%
```

When you create an archive using different options, always write the command syntax on the media label along with the names of the files or file system on the archive.

If you do not know which `cpio` options were used when an archive was created, all you can do is experiment with different combinations of the options to see which ones allow the archive to be read.

See the `cpio(1)` manual page for a complete list of options.

Retrieving Files Created With the bar Command

To retrieve files from diskettes that were archived using the SunOS 4.x `bar` command, use the `-H bar` option to `cpio`.

Note – You can only use the `-H bar` option with `-i` to retrieve files. You cannot create files with the `bar` header option.

▼ **How to Retrieve `bar` Files From a Diskette**

1. Change to the directory where you want to put the files.

2. Type `cpio -ivH bar < /vol/dev/rdiskette/unlabeled` and press Return.

All the files on the diskette are copied to the current directory.

The Cache File System

This chapter contains the following sections:

<i>Cache File System Concepts</i>	<i>page 79</i>
<i>Administering a Cache</i>	<i>page 81</i>
<i>Mounting a Cached File System</i>	<i>page 86</i>
<i>Checking a Cached File System</i>	<i>page 91</i>

Cache File System Concepts

A *cache* is a temporary storage area for data. The Cache File System (CacheFS) enables you to use disk drives on local workstations to store frequently used data from a remote file system or CD-ROM. The data stored on the local disk is the cache.

When a file system is cached, the data is read from the original file system and stored on the local disk. The reduction in network traffic improves performance and, if the remote file system is on a storage medium with slower response time than the local disk (such as CD-ROM), caching provides an additional performance gain.

CacheFS can use all or part of a local disk to store data from one or more remote file systems. A user accessing a file does not need to know whether the file is stored in a cache or is being read from the original file system. The user opens, reads, and writes files as usual.

The original file system, which may require network traffic to access, is called the *back file system* and files in it are *back files*. The *cached file system* resides on the local disk and files in it are *cached files*. The *cache directory* is a directory on the local disk where the data for the cached file system is stored. The file system in which the cache directory resides is called the *front file system* and its files are *front files*.

Resource Management

When you set up a cache, you can use all or part of an existing file system. You can also set up a new slice to be used by CacheFS.

In addition, when you create a cache, you can specify the percentage of resources, such as number of files or blocks, that CacheFS can use in the front file system. The configurable cache parameters are discussed in the section “Cache Parameters” on page 81.

Front File System Requirements

CacheFS requires a UFS file system for the front file system. The front file system can be on a metadvice, which is a grouping of two or more physical devices accessed as a single device.

You can use an existing UFS file system for the front file system or you can create a new one. Using an existing file system is the quickest way to set up a cache. Dedicating a file system exclusively to CacheFS gives you the greatest control over the file system space available for caching. For information on creating and mounting UFS file systems, see Chapter 3, “Mounting and Unmounting File Systems.”



Caution – Do not make the front file system read-only and do not set quotas on it. A read-only front file system prevents caching and file system quotas interfere with control mechanisms built into CacheFS.

Setting Up a Cached File System

There are two steps to setting up a cached file system:

1. You must create the cache with the `cfsadmin` command. See “Creating a Cache” on page 81.
2. You must mount the file system you want cached using the `-F cachefs` option to the `mount` command. See “Mounting a Cached File System” on page 86.

Administering a Cache

You use the `cfsadmin` command to perform the following tasks:

- Create a cache
- Set or modify cache parameters
- Display information about a cache and the cached file systems
- Delete a cache

The following sections describe how to perform each of these tasks.

Creating a Cache

To create a cache:

- ◆ **Type** `cfsadmin -c directory_name` and press **Return**.

The following example creates a cache and creates the cache directory `/local/mycache`. You must make sure the cache directory does not already exist.

```
# cfsadmin -c /local/mycache
```

This example uses the default cache parameter values. The CacheFS parameters are described in the section “Setting Cache Parameters” on page 83. See the `cfsadmin(1M)` manual page for more information on `cfsadmin` options.

Cache Parameters

The default values for the cache parameters are for a cache that uses the entire front file system for caching. To limit the cache to only a portion of the front file system, you should change the parameter values.

Table 5-1 shows the parameters for space and file allocation.

Table 5-1 CacheFS Parameters

Parameters for Space Allocation	Parameters for File Allocation
maxblocks	maxfiles
minblocks	minfiles
threshblocks	threshfiles

Table 5-2 shows the default values for the cache parameters. The default values for parameters devote the full resources of the front file system to caching.

Table 5-2 Default Values of Cache Parameters

Cache Parameter	Default Value
maxblocks	90 %
minblocks	0 %
threshblocks	85 %
maxfiles	90 %
minfiles	0 %
threshfiles	85 %

The `maxblocks` parameter sets the maximum number of blocks, expressed as a percentage, that CacheFS is allowed to claim within the front file system. The `maxfiles` parameter sets the maximum percentage of available inodes (number of files) CacheFS can claim.

Note - The `maxblocks` and `maxfiles` parameters do not guarantee the resources will be available for CacheFS—they set maximums. If you allow the front file system to be used for other purposes than CacheFS, there may be fewer blocks or files available to CacheFS than you intend.

The `minblocks` parameter does not guarantee availability of a minimum level of resources. The `minblocks` and `threshblocks` parameters work together. CacheFS can claim more than the percentage of blocks specified by `minblocks`

only if the percentage of available blocks in the front file system is greater than `threshblocks`. The `minfiles` and `threshfiles` parameters work together in the same fashion.

The `threshfiles` and `threshblocks` values apply to the entire front file system, not file systems you have cached under the front file system. The `threshblocks` and `threshfiles` values are ignored until the `minblocks` and `minfiles` values have been reached.

Note - Using the whole front file system solely for caching eliminates the need to change the `maxblocks`, `maxfiles`, `minblocks`, `minfiles`, `threshblocks`, or `threshfiles` parameters.

When the minimum, maximum, and threshold values are identical, CacheFS allows the cache to grow to the maximum size specified—if you have not reduced available resources by using part of the front file system for other storage purposes.

Setting Cache Parameters

To set parameters for a cache:

◆ **Type** `cfsadmin -c -o parameter_list cache_directory` and press **Return**.

The `parameter_list` has the following form:

`parameter_name1=value,parameter_name2=value,...`

The parameter names are listed in Table 5-1 on page 82. You must separate multiple arguments to the `-o` option with commas.

Note - The maximum size of the cache is by default 90% of the front file system resources. Performance deteriorates significantly if a UFS file system exceeds 90% capacity.

The following example creates a cache named `/local/cache1` that can use up to 80% of the disk blocks in the front file system and can grow to use 55% of the front file system blocks without restriction unless 60% (or more) of the front file system blocks are already in use.

```
# cfsadmin -c -o maxblocks=80,minblocks=55,threshblocks=60 /local/cache1
```

The following example creates a cache named `/local/cache2` that can use up to 75% of the files available in the front file system.

```
# cfsadmin -c -o maxfiles=75 /local/cache2
```

The following example creates a cache named `/local/cache3` that can use 75% of the blocks in the front file system, that can use 50% of the files in the front file system without restriction unless total file usage already exceeds 60%, and that has 70% of the files in the front file system as an absolute limit.

```
# cfsadmin -c -o maxblocks=75,minfiles=50,threshfiles=60,maxfiles=70 /local/cache3
```

Modifying Parameters

Note – Before changing parameters for a cache, you must unmount all file systems in the cache directory with the `umount` command.

To change the value of one or more parameters:

◆ **Type** `cfsadmin -u -o parameter_list cache_directory` and press **Return**.

Note – You can only increase the size of a cache, either by number of blocks or number of inodes. If you want to make a cache smaller, you must remove it and re-create it with new values.

The following example unmounts `/local/cache3` and changes the `threshfiles` parameter to 65 percent:

```
# umount mount_point
# cfsadmin -u -o threshfiles=65 /local/cache3
```

Displaying Information About Cached File Systems

To display information about all file systems cached under the specified cache directory:

◆ **Type** `cfsadmin -l cache_directory` and press Return.

The following example shows information about the cache directory named `/local/cache1`:

```
# cfsadmin -l /local/cache1
maxblocks      90%
  minblocks     90%
  threshblocks  90%
  maxfiles      90%
  minfiles      90%
  threshfiles   90%
  maxfilesize   3MB
mnt_c0t1d0s7
mnt1_man
#
```

In this example, two file systems are cached in `/local/cache1`. The last two lines display the cache IDs of the two cached file systems.

Deleting Cached File Systems

Note – Before deleting a cached file system, you must unmount all the cached files systems for that cache directory.

To delete a file system in a cache:

◆ **Type** `cfsadmin -d cache_id cache_directory` and press **Return**.

The cache ID is part of the information returned by `cfsadmin -l`. After deleting one or more of the cached file systems, you must run the `fsck_cachefs` command to correct the resource counts for the cache.

The following example unmounts a cached file system, deletes it from the cache, and runs `fsck_cachefs`.

```
# umount /usr/work
# cfsadmin -d _dev_dsk_c0t1d0s7 /local/cache1
# fsck -F cachefs /local/cache1
```

You can delete all file systems in a particular cache by using `all` as an argument to the `-d` option. The following command deletes all file systems cached under `/local/cache1`:

```
# cfsadmin -d all /local/cache1
```

The `all` argument to `-d` also deletes the specified cache directory.

Mounting a Cached File System

There are three ways to mount a file system in a cache:

- Using the `mount` command
- Creating an entry for the file system in the `/etc/vfstab` file
- Using `autofs`

See *NFS Administration Guide* for more information on mounting a cached file system with `autofs`.

Using `mount` to Mount a Cached File System

To mount a file system in a cache:

◆ **Type** `mount -F cachefs -o backfstype=type, cachedir=cache_directory back_filesystem mount_point` and press **Return**.

The arguments used with the `-o` option are described in Table 5-3 on page 89. See the `mount_cachefs(1M)` manual page for more information about the arguments used when mounting a cached file system.

For example, the following command makes the file system `merlin:/docs` available as a cached file system named `/docs`:

```
# mount -F cachefs -o backfstype=nfs,cachedir=/local/cache1 merlin:/docs /docs
```

Mounting a Cached File System That Is Already Mounted

You use the `backpath` argument when the file system you want to cache has already been mounted. `backpath` specifies the mount point of the mounted file system. When the `backpath` argument is used, the back file system must be read-only. If you want to write to the back file system, you must unmount it before mounting it as a cached file system.

For example, if the file system `merlin:/doc` is already NFS-mounted on `/nfsdocs`, you can cache that file system by giving that path name as the argument to `backpath`, as shown in the following example:

```
# mount -F cachefs -o backfstype=nfs,cachedir=/local/cache1,backpath=/nfsdocs merlin:/doc /doc
```

Note – There is no performance gain in caching a local UFS disk file system.

Mounting a CD-ROM as a Cached File System

So far, examples have illustrated back file systems that are NFS-mounted, and the device argument to the `mount` command has taken the form `server:file_system`. If the back file system is a High Sierra or Rockridge file system, the device argument is the CD-ROM device in the `/cdrom` directory. The file system type is either `hsfs` or `rockridge`.

The following example illustrates caching a High Sierra back file system on the device `/cdrom/cdrom_name` as `/doc` in the cache `/local/cache1`:

```
# mount -F cachefs -o backfstype=hsfs,cachedir=/local/cache1,ro, backpath=/cdrom/cdrom_name \  
/cdrom/cdrom_name /doc
```

Because you cannot write to the CD-ROM, the `ro` argument is specified to make the cached file system read-only. The arguments to the `-o` option are explained in Table 5-3.

You must specify the `backpath` argument because Volume Management automatically mounts the CD-ROM when it is inserted. The mount point is in the `/cdrom` directory and is determined by the name of the CD-ROM. The special device to mount is the same as the value for the `backpath` argument.

Consistency Checking

To ensure that the cached directories and files are kept up to date, CacheFS periodically checks consistency of files stored in the cache. To check consistency, CacheFS compares the current modification time to the previous modification time; if the modification times are different, all data and attributes for the directory or file are purged from the cache and new data and attributes are retrieved from the back file system.

When an operation on a directory or file is requested, CacheFS checks to see if it is time to verify consistency. If so, CacheFS obtains the modification time from the back file system and performs the comparison.

CacheFS calculates the next time to perform a consistency check for directories as follows:

$$\begin{aligned} X &= \text{time-of-day} - \text{current-modification-time} \\ Y &= \text{MAX}(X, \text{acdirmin}) \\ Z &= \text{MIN}(Y, \text{acdirmax}) \\ \text{time-of-next-check} &= Z + \text{time-of-day} \end{aligned}$$

The same calculation is performed to determine the next time to check consistency for files, except that `acregmin` is substituted for `acdirmin` in the MAX calculation and `acregmax` is substituted for `acdirmax` in the MIN calculation.

The default value for `acdirmin`, `acdirmax`, `acregmin`, and `acregmax` is 30 seconds. `acdirmin`, `acdirmax`, `acregmin`, and `acregmax` are parameters to the `mount` command and can be changed when the file system is mounted. See Table 5-3 for more information on `mount` parameters.

Additional mount Arguments

Table 5-3 describes the arguments that can be used with the `-o` option of `mount`.

Table 5-3 Arguments Used With the `-o` mount Option

Parameter	Description
<code>acdirmax=n</code>	Used in determining the next time to consistency. See “Consistency Checking” on page 88. The default is 30 seconds.
<code>acdirmin=n</code>	Used in determining the next time to consistency. See “Consistency Checking” on page 88. The default is 30 seconds.
<code>acregmax=n</code>	Used in determining the next time to consistency. See “Consistency Checking” on page 88. The default is 30 seconds.
<code>acregmin=n</code>	Used in determining the next time to consistency. See “Consistency Checking” on page 88. The default is 30 seconds.
<code>actimeo=n</code>	Sets <code>acregmin</code> , <code>acregmax</code> , <code>acdirmin</code> , and <code>acdirmax</code> to <i>n</i> .
<code>backfstype</code>	Specifies the file system type of the back file system.
<code>backpath</code>	Specifies the mount point of the back file system. Use only if the back file system is already mounted.
<code>cachedir</code>	Specifies the name of the cache directory.
<code>cacheid</code>	Allows you to assign a string to identify each separate cached file system. If you do not specify a <code>cacheid</code> , CacheFS generates one. You need the <code>cacheid</code> when you delete a cached file system with <code>cfsadmin -d</code> . A <code>cacheid</code> you choose will be easier to remember than one automatically generated. The <code>cfsadmin</code> command with the <code>-l</code> option includes the <code>cacheid</code> in its display.

Table 5-3 Arguments Used With the `-o mount` Option

Parameter	Description
<code>local-access</code>	Improves performance by having CacheFS check the mode bits. By default, the back file system interprets the mode bits used for access checking to ensure data integrity. Do not use this argument if you are running secure NFS.
<code>noconst</code>	Disables consistency checking between the front and back file systems. Use <code>noconst</code> when the back file system and cache file system are read-only. Otherwise, always allow consistency checking. The default is to enable consistency checking. If none of the files in the back file system will be modified, you can use the <code>noconst</code> option to <code>mount</code> when mounting the cached file system. Changes to the back file system may not be reflected in the cached file system.
<code>rw</code> <code>ro</code>	Make the cached file system read-write or read-only. The default is read-write.
<code>suid</code> <code>nosuid</code>	Allow or disallow setuid execution. The <code>suid</code> argument is the default. For an explanation of setuid execution, see <i>Security, Performance, and Accounting Administration</i> .
<code>write-around</code> <code>non-shared</code>	Prevents inconsistencies resulting from someone else writing to the back file system while a user is working with the same files in a cache. In this mode, each write is made directly to the back file, and the front file is purged from the cache. You bypass, or “write around,” the cache. The file can be brought back into the cache, but writing directly to the back file ensures that no one opens a file that is out-of-date because it was changed while it was in the cache. <code>write-around</code> mode also protects against the reverse situation: opening a cached file right before someone writes to the back file and thus starting with an out-of-date version. Like NFS, CacheFS enforces consistency by checking at regular intervals to see if any cached file has been changed in the back file system. If it has, it is purged from the cache, so that the next time a user accesses the file, the current version is retrieved from the back file system. <code>write-around</code> mode is the default. Use <code>non-shared</code> mode when only one user at a time is writing to a particular set of files in the cache. Because it is assumed that no one else will write concurrently to the same file, or concurrently modify directories, writes are made to both the front and back files, and the files the user is working with remain active in the cache.

Creating a `vfstab` Entry for Cached File Systems

As with other file system types, you can put an entry in the `/etc/vfstab` file for a cached file system to mount the cached file system automatically every time the system boots. The `/etc/vfstab` file has the following seven fields:

- Device to mount
- Device to `fsck`
- Mount point
- File system type
- `fsck` pass
- Mount at boot
- Mount options

Enter the special device name of the back file system as the device to mount. For NFS file systems, the entry takes the form `server:path`. The device to `fsck` is the cache directory path. The mount point is the mount point of the cached file system. The following example shows an entry for a cached file system (the lines beginning with hash marks (#) are comments):

#device	device	mount	FS	fsck	mount	mount
#to mount	to fsck	point	type	pass	at boot	options
server1:/usr/abc	/cache1	/mydocs	cachefs2	yes		rw,backfstype=nfs,cachedir=/cache1

See “The Virtual File System Table” on page 26 for more information on `vfstab` entries.

Checking a Cached File System

The `fsck` command is used to check the integrity of cached file systems. The CacheFS version of `fsck` automatically corrects problems without requiring user interaction.

To check a cached file system:

- ◆ **Type `fsck -F cachefs cache_directory` and press Return.**

The following example checks the cache directory `/local/cache1`:

```
# fsck -F cachefs /local/cache1
```

You should not need to run `fsck` manually for cached file systems; `fsck` is run automatically at boot time or when the file system is mounted.

Two options are available for the CacheFS version of `fsck`: `-m` and `-o noclean`. The `-m` option causes `fsck` to check the specified file system without making any repairs. The `-o noclean` option forces a check of the file system. See the `fsck_cacheofs(1M)` man page for more information.

Part 2 — Backup and Restore


This part has three chapters:

Chapter 6, “Planning a Backup Strategy,” describes why you need a backup strategy, explains the `ufsdump` command and how it works, and describes how to choose which file systems to back up, which media to use, and how to plan a backup schedule.

Chapter 7, “Backing Up Files and File Systems,” provides steps for using the `ufsdump` command to back up files and file systems.

Chapter 8, “Restoring Files and File Systems,” provides steps for using the `ufsrestore` command to restore files and file systems.

Planning a Backup Strategy

6 

This chapter contains these sections:

<i>Why You Back Up File Systems</i>	<i>page 95</i>
<i>Understanding the ufsdump Command</i>	<i>page 96</i>
<i>Choosing Which File Systems to Back Up</i>	<i>page 99</i>
<i>Choosing Which Media to Use</i>	<i>page 102</i>
<i>Considering Other Issues</i>	<i>page 108</i>
<i>Planning a Backup Schedule</i>	<i>page 112</i>

Why You Back Up File Systems

Backing up files means making copies of them, usually on removable media, as a safeguard in case the originals get lost or damaged. Backup tapes are convenient for restoring accidentally deleted files, but they are essential in case of serious hardware failures or other disasters.

Backing up files is one of the most crucial system administration functions. You must plan and carry out a procedure for regularly scheduled backups of your file systems for three major reasons:

- To ensure file system integrity against a possible system crash
- To ensure user files against accidental deletion
- To act as an important safeguard before reinstalling or upgrading a system

When you back up file systems as scheduled, you have the assurance that you can restore files to a reasonably recent state. In addition, you may want to back up file systems to transport them from one system to another or to *archive* them, saving files on a transportable media, so that you can remove or alter the files that remain on the system.

When you plan a backup schedule, you need to consider:

- Which command to use to back up the file systems
- Which file systems to back up
- What media to use
- What backup schedule to use

Understanding the `ufsdump` Command

You usually use the `ufsdump` command to do periodic backups in which you save all the files recently modified on a specified file system. This section and Chapter 7, “Backing Up Files and File Systems,” describe the `ufsdump` command. See Chapter 4, “Copying UFS Files and File Systems,” for a description of `dd`, `tar`, `cpio`, and other commands that can be used to copy files.

Networker for Solaris also provides backup and restore capabilities. See the *Networker for Solaris User's Guide* for information about backing up client machines. See the *Networker for Solaris Administrator's Guide* for information about configuring Networker for Solaris and backing up servers.

Advantages of `ufsdump`

The `ufsdump` command is designed to back up entire file systems. You can also use it to back up individual files.

The `ufsdump` command has these advantages over other methods for performing backups:

- It provides incremental backups. You can specify different backup (dump) levels, making it possible to back up only those files that were changed since a previous backup at a lower level.
- It works quickly. The command knows the structure of the UFS file system type and works directly through the raw device file.

- It supports multiple volumes. You can back up very large file systems by writing the backup to multiple volumes (usually tapes).
- It supports remote drives. The media drive can be on any system in the network to which the user has access. High-capacity tape drives can be used to back up many systems.

Disadvantages of `ufsdump`

`ufsdump` cannot automatically calculate the number of tapes or diskettes needed for a backup. You can, however, use the `S` option to determine the amount of space that is needed to perform the dump (without actually doing it) and display the estimated number of bytes it will take.

The `ufsdump` command does not have built-in error-checking to minimize problems if you are backing up an active file system.

How `ufsdump` Works

Basic Operation

The `ufsdump` command makes two passes when backing up a file system. On the first pass, it scans the raw device file for the file system and builds a table of directories and files in memory. It then writes the table to the backup media. In the second pass, `ufsdump` goes through the inodes in numerical order, reading the file contents and writing it to the media.

Determining Device Characteristics

The `ufsdump` command needs to know only an appropriate block size and how to detect the end-of-media.

Detecting the End of Media

`ufsdump` writes a sequence of fixed-size records. When `ufsdump` receives notification that a record was only partially written, it assumes that it has reached the physical end of the media. This method works for most devices. If a device is not able to notify `ufsdump` that only a partial record has been

written, a media error occurs as `ufsdump` tries to write. DAT devices and 8mm tape devices detect end-of-media. Cartridge tape devices and 1/2 inch tape devices do not detect end-of-media.

Copying Data

The `ufsdump` command copies data only from the raw disk slice. If the file system is still active, anything in memory buffers is probably not copied. The backup done by `ufsdump` does not copy free blocks, nor does it make an image of the disk slice. If symbolic links point to files on other slices, the link itself is copied.

The Role of /etc/dumpdates

The `ufsdump` command, when used with the `u` option, maintains and updates a file named `/etc/dumpdates`. Each line in `/etc/dumpdates` shows the file system backed up, the level of the last backup, and the day, date, and time of the backup. Here is a typical `/etc/dumpdates` file from a file server:

```

/dev/rdsk/c0t1d0s0 0 Fri Nov 6 07:54:38 1993
/dev/rdsk/c0t1d0s5 0 Sat Oct 10 07:53:44 1993
/dev/rdsk/c0t1d0s7 0 Sat Oct 10 07:56:57 1993
/dev/rdsk/c0t1d0s6 0 Sat May 23 08:02:34 1993
/dev/rdsk/c0t1d0s0 5 Fri Nov 6 07:55:20 1993
/dev/rdsk/c0t1d0s7 5 Fri Nov 6 07:58:08 1993
/dev/rdsk/c0t1d0s6 5 Fri May 29 09:03:07 1993
/dev/rdsk/c0t1d0s5 9 Thu Nov 5 07:15:51 1993
/dev/rdsk/c0t1d0s4 9 Thu Nov 5 07:18:04 1993
/dev/rdsk/c0t1d0s6 9 Thu Jun 4 09:21:02 1993
```

When you do an incremental backup, the `ufsdump` command consults `/etc/dumpdates` to find the date of the most recent backup of the next lower level. Then it copies to the media all files that were updated since the date of that lower-level backup. After the backup is complete, a new information line, describing the backup you just completed, replaces the information line for the previous backup at that level. On the date that you do a level 0 backup, `/etc/dumpdates` contains one information line for each backed up file system at each level.

Use the `/etc/dumpdates` file to verify that backups are being done. This verification is particularly important if you are having equipment problems. If a backup cannot be completed because of equipment failure, the backup is not recorded in the `/etc/dumpdates` file.

If you need to restore an entire disk, check the `/etc/dumpdates` file for a list of the most recent dates and levels of backups so that you can determine which tapes you need to restore the entire file system.

Note - `/etc/dumpdates` is a text file that can be edited, but edit it only at your own risk. If you make changes to the file that do not match your archive tapes, you may not be able to find the tapes (or files) you need.

Choosing Which File Systems to Back Up

Consider these factors as an important part of planning your backup strategy:

- What files are critical to users on this system?
- Where are the files located? Are they in a single file system?
- How often do these files change?
- How quickly would you need to restore these files in the event of damage or loss?
- How often can the relevant file systems be unmounted so that they are available for backup?

The next sections describe some additional factors to consider in planning your backup strategy.

You need to back up file systems that change frequently. The file systems that need to be backed up for a given system depend on the type of system.

File Systems to Back Up on a Standalone System

By default, the installation procedure creates at least the file systems shown in Table 6-1.

Table 6-1 Default File Systems on a Standalone System

Directory	Partition
/	0
/usr	6

The / file system on a standalone system contains /kernel/unix and other important files. It also contains the /var directory, in which frequently modified files, such as mail and accounting, are kept. Therefore, you should back up the / file system at regular intervals.

Back up the /usr file system occasionally, especially if you install new software or add new commands.

/export/home contains the directories and subdirectories of all the users on the standalone system. Back up /export/home more often than / or /usr, perhaps as often as once a day, depending on your site's requirements.

During installation, you may have assigned file systems such as /export or /var to other available slices.

Be aware of the slices where file systems are located. Use the df command or look at the /etc/vfstab file to find out on which slice a file system is located.

File Systems to Back Up on a Server

On a file server, you have to back up not only the file systems that contain the operating system itself, but the file systems for the individual users as well.

The default file systems on a server are as shown in Table 6-2.

Table 6-2 Default File Systems on a Server

Directory	Slice
/	0
/usr	6
/export/home	7
/export/swap	4
/export	3

Periodically, you need to back up the file systems containing the kernel, major commands, and executables (`/`, `/usr`, and `/export`). Back up these file systems at intervals from once a day to once a month, depending on your site's requirements. For example, if you frequently add and remove clients and equipment on the network, you have to change important files in root, including the kernel configuration file. In this case, you might want to back up root more frequently than if your site seldom changes the network configuration. Furthermore, your site may keep users' mail in the directory `/var/mail` on a mail server, which client machines then mount. If that is the case, you might want to back up root daily to preserve mail. `/usr` contents are fairly static and only need to be backed up from once a week to once a month.

The root directory of diskless clients is kept in the `/export` file system. Because the information it contains is similar to the server's root directory in slice 0, it does not change frequently. If your site sends mail to clients' root directories, you should back up `/export` more frequently.

You need not back up `/export/swap`.

The file system you need to back up the most frequently is `/export/home`. Because `/export/home` contains the home directories and subdirectories of all the users on the system, its files are very volatile. Depending on your site, you should back up `/export/home` at least once a week, if not daily.

Choosing Which Media to Use

You typically back up Solaris systems using 1/2-inch reel tape, 1/4-inch streaming cartridge tape, 8-mm cartridge tapes, or 4-mm cartridge tapes (DAT). You can perform backups using diskettes, but this is time-consuming and cumbersome.

The media you choose depends on the availability of the equipment that supports it and of the media (usually tape) that you use to store the files. Although you must do the backup from a local system, you can write the files to a remote device. Table 6-3 shows typical media used for backing up file systems and shows the length (or storage capacity) for each. You can use the storage capacity as input for the `-s size` option to the `ufsdump` command.

Table 6-3 Media Storage Capacities

Media	Capacity	Tape Length
1/2-inch tape	40–45 Mbytes	2300 feet
60-Mbyte 1/4-inch cartridge	60 Mbytes	425 feet
150-Mbyte 1/4-inch cartridge	150 Mbytes	700 feet
2.3-Gbyte 8-mm	2.3 Gbytes	6000 feet
5.0-Gbyte 8-mm	5.0 Gbytes	13000 feet
3.5-inch diskette	1422 blocks (1.44 Mbytes)	

Note – Capacity for 4-mm cartridge tapes is dependent on the type of drive and the data being written to the tape.

Backup Device Names

You specify a tape or diskette drive to use for backup by supplying a logical device name. This name points to the subdirectory containing the “raw” device file and includes the logical unit number of the drive. Table 6-4 shows this naming scheme.

Table 6-4 Basic Device Names for Backup Devices

Device Type	Name
Tape	<code>/dev/rmt/<i>unit</i></code>
Diskette	<code>/vol/dev/rdiskette0/unlabeled</code>

The drive writes at its “preferred” density, which usually means the highest density it supports. Most SCSI drives can automatically sense the density or format on the tape and read it accordingly.

Tape drive naming conventions use a logical, not a physical, device name.

Tape drives fall into two categories according to controller type:

- Xylogics 472 for 1/2-inch rack-mounted (top-loaded) reel-to-reel drives (maximum 4 units per controller)
- SCSI for 1/4-inch cartridge, 1/2-inch front-loaded reel-to-reel, and 4-mm or 8-mm helical scan drives (maximum 7 units per controller)

Within the `/dev/rmt` subdirectory is a single set of tape device files that support different output densities.

In general, you specify a tape drive device as shown in Figure 6-1.

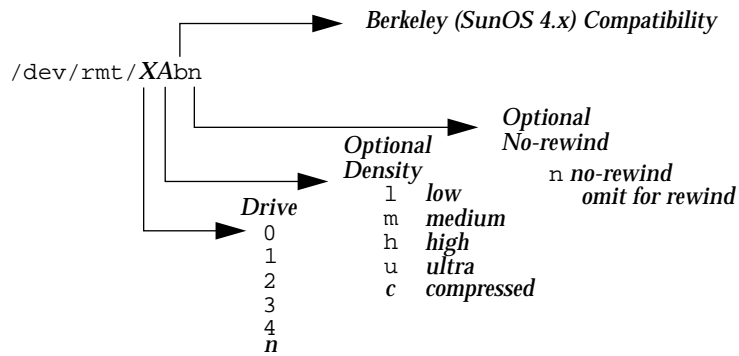


Figure 6-1 Tape Drive Device Names

▼ How to Specify the Default Density for a Tape Drive

Normally, you specify a tape drive by its logical unit number, which may run from 0 to *n*. For example, to specify the first drive, rewinding, use:

```
/dev/rmt/0
```

To specify the first drive, nonrewinding, use:

```
/dev/rmt/0n
```

To specify the second drive, rewinding, use:

```
/dev/rmt/1
```

To specify the second drive, nonrewinding, use:

```
/dev/rmt/1n
```

By default, the drive writes at its “preferred” density, which is usually the highest density it supports. If you do not specify a tape device, the command writes to drive number 0 at the default density the device supports.

▼ How to Specify Different Densities for a Tape Drive

You may want to transport a tape to a system whose tape drive supports only a certain density. In that case, specify a device name that writes at the desired density. Use this convention to specify rewind:

```
/dev/rmt/XA
```

and this convention to specify no rewind:

```
/dev/rmt/XAn
```

The unit and density characters are shown in Figure 6-1.

For example, to specify a raw magnetic tape device on the first (0) drive with medium density and no rewinding, use:

```
/dev/rmt/0mn
```

You can have both SCSI and non-SCSI tape drives on the same system. A SCSI controller can have a maximum of seven SCSI tape drives, and a non-SCSI controller can have a maximum of four tape drives. For each drive number (X), the density character depends on the controller and drive type as described in the following paragraphs.

Table 6-5 shows the device abbreviation for different tape controllers/units and media. Note that the first character in the device abbreviation for drive number does not have to be 0 as shown, but could be 1, 2, or 3, and so on, depending on how many tape drives are attached to the system.

Table 6-5 Device Abbreviations for Tape Controllers/Units and Media

Controller	Drive Unit	Size	Type	Format	Tracks	Device Abbreviation
Xylogics® 472	Fujitsu M2444	1/2-inch	Reel	1600 bpi	9	/dev/rmt/0m
		1/2-inch	Reel	6250 bpi	9	/dev/rmt/0h
SCSI front-loaded	HP	1/2-inch	Reel	800 bpi	9	/dev/rmt/0m
				6250 bpi	9	/dev/rmt/0h
SCSI	Sysgen®	1/4-inch	Cartridge	QIC-11	4	/dev/rmt/0l
				QIC-24	4	/dev/rmt/0m
				QIC-11	9	/dev/rmt/0l
				QIC-24	9	/dev/rmt/0m
	Emulex® MT-02	1/4-inch	Cartridge	QIC-11	4	/dev/rmt/0l
				QIC-24	4	/dev/rmt/0m
				QIC-11	9	/dev/rmt/0l
				QIC-24	9	/dev/rmt/0m
Archive® QIC-150	1/4-inch	Cartridge	QIC-150	18	/dev/rmt/0h	

Table 6-5 Device Abbreviations for Tape Controllers/Units and Media (Continued)

Controller	Drive Unit	Size	Type	Format	Tracks	Device Abbreviation
	Wangtek™ QIC-150	1/4-inch	Cartridge	QIC-150	18	/dev/rmt/0h
	Desktop Backup Pack	1/4-inch	Cartridge	QIC-150	18	/dev/rmt/0h
	Exabyte® 8200 (2.3 GB)	8 mm	Cartridge	8 mm	Helical Scan	/dev/rmt/0m
	Exabyte 8500 (2.3 GB)	8 mm	Cartridge	8 mm	Helical Scan	/dev/rmt/0l
	Exabyte 8500 (5 GB)	8 mm	Cartridge	8 mm	Helical Scan	/dev/rmt/0m
	Archive® Python	4 mm	Cartridge	4 mm	Helical Scan	/dev/rmt/0

Rack-Mounted Non-SCSI 1/2-Inch Reel Drives

For 1/2-inch rack-mounted tape drives with either a Tapemaster or Xylogics 472 controller, substitute the density from Table 6-6 for the A variable in the device name (/dev/rmt/XA).

Table 6-6 Designating Density for Rack-Mounted 1/2-inch Tape Drives

Character	Density
null	Default “preferred” (highest) density (usually 6250 bpi uncompressed)
l	800 bpi
m	1600 bpi
h	6250 bpi
u	6250 bpi compressed

If you omit the density character, the tape is usually written at its highest density, not compressed.

SCSI 1/4-Inch Cartridge and 1/2-Inch Front-Loaded Reel Drives

For SCSI 1/4-inch cartridge and 1/2-inch front-loaded reel drives, substitute the density from Table 6-7 for the A variable in the device name (`/dev/rmt/XA`).

Table 6-7 Designating Format or Density for SCSI Tape Drives

Character	Density 1/4-Inch Cartridge	Density 1/2-Inch Front-Loaded Reel-to-Reel
null	Default preferred (highest) density	Default preferred (highest) density
l	QIC-11 format	800 bpi
m	QIC-24 format	1600 bpi
h	QIC-150	6250 bpi
u	Reserved	Reserved

For 1/4-inch cartridges, density is specified by the format in which the data is written: the QIC format. The QIC-11 and QIC-24 format write approximately 1000 bytes per inch on each track. The density for QIC-150 is somewhat higher. The “preferred” density for a 60-Mbyte 1/4-inch cartridge drive is QIC-24 and for a 150-Mbyte 1/4-inch cartridge drive is QIC-150.

An 18-track drive can write only QIC-150; it cannot be switched to write QIC-24 or QIC-11. Format selection is only useful for drives that can write both QIC-24 and QIC-11.

Guidelines for Drive Maintenance and Media Handling

A backup tape that cannot be read is useless. It is a good idea to clean and check your tape drives periodically to ensure correct operation. See your hardware manuals for instructions on procedures for cleaning a tape drive. You can test your tape hardware by copying some files to the tape and reading them back and then comparing the original with the copy. Or you could use the `v` option of the `ufsdump` command to verify the contents of the media with the source file system. The file system must be unmounted or completely idle for the `v` option to be effective. Be aware that hardware can fail in ways that the system does not report.

Always label your tapes after a backup. If you have planned a backup strategy similar to those suggested in the next section, you should indicate on the label “Tape A,” “Tape B,” and so forth. This label should never change. Every time you do a backup, make another tape label containing the backup date, the name of the machine and file system backed up, backup level, the tape number (1 of *n*, if it spans multiple volumes), plus any information specific to your site. Store your tapes in a dust-free safe location, away from magnetic equipment. Some sites store archived tapes in fireproof cabinets at remote locations.

You need to create and maintain a log that tracks which media (tape volume) stores each job (backup) and the location of each backed-up file.

Considering Other Issues

This section describes these other issues to consider for determining a backup strategy for your site:

- “When to Run Backups” on page 108
- “How Long to Save Backups” on page 109
- “How to Back Up Files to a Remote Drive” on page 109
- “Do You Need to Become Superuser?” on page 110
- “Should You Check File Systems Before a Full Backup?” on page 110
- “Do You Want to Put Multiple Backups on the Same Tape?” on page 111
- “Where Do the Files Reside?” on page 111
- “How Do You Backup on a Heterogeneous Network?” on page 111

When to Run Backups

In deciding when to run backups, consider things such as availability of an operator, impact on system performance, minimizing data loss, and level of file system activity.

Although operators may be less available, doing backups during off hours minimizes the impact on system performance and decreases the likelihood of file system activity getting in the way. The sooner you back up files after they have been changed, the less chance there is of loss of data.

Traditionally, operator intervention was important—someone had to change the tapes when they filled up. It is becoming less important because of higher-capacity and autoloading drives.

You can automate off-hours backups by having the `crontab` utility call a script that starts the `ufsdump` command. Whatever time you decide on, consistently run your backups at the same time each day.

How Long to Save Backups

Depending on how many file systems you back up, their size, and the capacity of your media, you may use quite a few tapes. The longer you save the tapes (rather than reusing them), the more tapes you need. You should be able to restore files for the last four weeks. Thus you should have at least four sets of tapes, one for each week, and rotate them each month. In addition, you should archive the full backups done monthly for at least a year, and then keep a yearly backup for a number of years.

Requirements for backups vary from site to site. You may want to be able to restore files from the last three months instead of four weeks. You may also need to do full backups weekly instead of monthly and keep the full backups for more than a year. The number of tapes you require will vary depending on your backup schedule, the size of the file systems, and the amount of activity on the file systems.

How to Back Up Files to a Remote Drive

To back up a set of systems over the network, you can run the `ufsdump` command from one system on each remote system (through remote shell or remote login) and direct the output to the system on which the drive is located. Note that you cannot directly specify files on remote systems in the `files-to-backup` argument. Typically, the drive is located on the system from which you run the `ufsdump` command, but it does not have to be.

You can use the `ufsdump` command to access a remote drive over the network. The command syntax is:

```
ufsdump [ options ] remote-host : backup-file files-to-backup
```

The only difference between this command and one you would use when backing up to a local drive is that the *backup-file* argument is prefaced with the *remote-host* name: the name of the remote system with the tape drive you want to back up to. Be sure to include the colon (:) after the host name, with no spaces before or after it.

Note – The naming convention you use for the remote drive depends on the system where the drive resides. If the drive is on a system that is running SunOS 4.x, use the SunOS 4.x convention (for example, `/dev/rst0`). If the system is running SunOS 5.x system software, use the SunOS convention (for example, `/dev/rmt/0m`).

To run a backup as a different user on a remote host, the command syntax is as follows:

```
ufsdump [options] user@remote-host:backup-file files-to-backup
```

Another way to back up files to a remote drive is to pipe the output from `ufsdump` to the `dd` command. See Chapter 4, “Copying UFS Files and File Systems,” for information about using the `dd` command.

To be able to do backups across the network, you need access to the systems. You have to put entries in `/.rhosts` files on both clients and server to do centralized backups. If you have appropriate permissions, edit the remote system’s `/.rhosts` file. Add the system from which you back up to the list of trusted hosts. Thereafter, you can log in as superuser on that system and do a backup to the remote system. If you do not have the proper permissions for the remote system, ask the system administrator to edit `/.rhosts` for you.

Do You Need to Become Superuser?

Because `ufsdump` needs to have read access to the raw device files, you should become superuser or be a member of the `sys` group to run the `ufsdump` command. Giving ordinary users read permissions for raw device files is a potential security problem.

Should You Check File Systems Before a Full Backup?

Most of the time, you do not need to check file systems for consistency before doing a backup. If you suspect a file system problem, you probably should run a consistency check. You can use the `-m` option to the `fsck` command to see if a file system needs further consistency checking. See Chapter 12, “Checking File System Integrity,” for information about using the `fsck` command.

Do You Want to Put Multiple Backups on the Same Tape?

If you are backing up multiple file systems that do not fill up a whole tape, you can use the “no rewind” option and place one file system after the next on a single tape. Putting multiple file systems on one tape can be quite useful for incremental backups, which may take up much less space than full backups, and it reduces the number of tapes needed. Keep in mind, however, that putting multiple file system backups on a single tape complicates finding the version of a file you want to restore.

Where Do the Files Reside?

You can back up local file systems only. Many of the file systems available to a workstation are mounted across the network from a server. Consequently, they reside on a server, not the workstation. For example, the `/export/home` directories for workstation users typically are located on servers. These file systems are backed up from the server, not the workstation. In fact, users are denied permission if they try to run `ufsdump` on files they own that are located on a server.

How Do You Backup on a Heterogeneous Network?

The command used to back up a file system has to be run on the local file system. You can, however, use the `rlogin` or `rsh` command to log onto the system. You can also direct the output of the `ufsdump` command to a remote tape drive.

When doing backups on a heterogeneous network, always use the device naming convention appropriate for the individual system. For example, run the SunOS 4.x `dump` command on a SunOS 4.x client system. Remember to use that SunOS 4.x device name (for example, `/dev/sd0h`).

The naming convention you use for the remote drive depends on the system to which the drive is attached. If the drive is on a system that is running SunOS 4.x, use the SunOS 4.x device name (`/dev/rst0`). If the system is running SunOS 5.x, use the SunOS 5.x device name (for example, `/dev/rmt/0m`).

To back up a SunOS 5.x client from a SunOS 4.x server, use the `rlogin` (or `rsh`) command and run `ufsdump` on the client system. If the tape drive is on a SunOS 5.x client, use the SunOS 5.x tape device name (for example, `/dev/rmt/0m`).

What Are the Security Issues?

The `ufsdump` and `ufsrestore` commands respect restricted access to back up devices and target file systems.

If you want to be cautious, you should:

- Set restrictive file permissions on any backup table of contents you create and store on the system (see the `a` option to `ufsdump`).
- Be sure the permissions on raw device files for disk slices are set so only privileged operators can back them up.
- Be aware that providing entries in `/.rhost` files on both clients and servers for centralized backups can be a possible security hole for unauthorized access.

Planning a Backup Schedule

You can use the `ufsdump` command to do two kinds of backups: *full backups* and *incremental backups*. A full backup includes all the files in the specified file system or directory. An incremental backup includes only those files in the specified file system that have changed since a previous backup at a *lower* level. The level of the backup determines which files are backed up.

Level 0 backs up the complete file system. Levels 1 through 9 perform incremental backups. Whenever you do a backup, you can tell the `ufsdump` command the level of the backup to perform. If you do not specify a level, the command defaults to level 9. A *backup schedule* is the schedule you establish to run the `ufsdump` command on a regular basis at different levels. You run the `ufsdump` command and specify the level of dump that matches the schedule you have chosen.

The Outcomes of Different Backup Schedules

The following sections discuss some possible schedules. All schedules assume you begin with a full backup (level 0) and that you use the `u` option to record each backup in the `/etc/dumpdates` file.

The Nine-to-Five

The schedule shown in Table 6-8 is probably the most commonly used, and is recommended for most situations.

Table 6-8 Daily Cumulative/Weekly Cumulative Backup Schedule

	Floating	Mon	Tues	Wed	Th	Fri
1st of Month	0					
Week1		9	9	9	9	5
Week2		9	9	9	9	5
Week3		9	9	9	9	5
Week4		9	9	9	9	5

With this schedule, each weekday tape accumulates all files changed since the end of the previous week (or the initial level 0 for the first week) and each Friday's tape contains all the files changed since the first level 0. For the level 9 backups, the previous level 0 or level 5 is the closest backup at a lower level. All the files that have changed since that lower-level backup at the end of the previous week are saved each day. For the Friday level 5, the nearest lower-level backup is the level 0 done at the beginning of the month. Consequently, each Friday's tape contains all the files changed during the month to that point.

Table 6-9 shows how the contents of the tapes can change across two weeks.

Table 6-9 Contents of Tapes for Daily/Weekly Cumulative Schedule

	Mon	Tues	Wed	Th	Fri
Week1	a b	a b c	a b c d	a b c d e	a b c d e f
Week2	g	g h	g h i	g h i j	a b c d e f g h i j k

With this schedule, you need a minimum of six or nine tapes: one for the level 0, four for the Fridays, and one daily tape (if it is reused) or four daily tapes. Use four daily tapes to save different versions of files. When you reuse the daily tape, only the latest version of a file (during that week) is saved. If you use different daily tapes and save the daily tapes for four weeks before reusing them, which is recommended, the total needed is 21.

If you need to restore the complete file system you need three tapes: the level 0, the most recent Friday tape, and the most recent daily tape.

The Nine- to-Two-Three-Four-Five

Table 6-10 shows a schedule where each weekday tape accumulates all files changed since the beginning of the week (or the initial level 0 for the first week) and each Friday's tape contains all the files changed that week.

Table 6-10 Daily Cumulative/Weekly Incremental Backup Schedule

	Floating	Mon	Tues	Wed	Th	Fri
1st of Month	0					
Week1		9	9	9	9	3
Week2		9	9	9	9	3
Week3		9	9	9	9	4
Week4		9	9	9	9	5

Table 6-11 shows how the contents of the tapes can change across two weeks.

Table 6-11 Contents of Tapes for Daily Cumulative/Weekly Incremental Schedule

	Mon	Tues	Wed	Th	Fri
Week1	a b	a b c	a b c d	a b c d e	a b c d e f
Week2	g	g h	g h i	g h i j	g h i j k

With this schedule, you need at least six or nine tapes: one for the level 0, four for the Fridays, and one daily tape (if reused) or four daily tapes, assuming you reuse the daily tapes each week.

If you need to restore a complete file system, you need five tapes: the level 0, all preceding Friday tapes, and the most recent daily tape.

The Three-Four-Five-Six-to-Two

Table 6-12 shows a schedule where each weekday tape contains only the files changed since the previous day and each Friday's tape contains all files changed since the initial level 0 at the beginning of the month.

Table 6-12 Daily Incremental/Weekly Cumulative Backup Schedule

	Floating	Mon	Tues	Wed	Th	Fri
1st of Month	0					
Week2		3	4	5	6	2
Week3		3	4	5	6	2
Week4		3	4	5	6	2

Table 6-13 shows how the contents of the tapes can change across two weeks.

Table 6-13 Contents of Tapes for Daily/Weekly Cumulative Schedule

	Mon	Tues	Wed	Th	Fri
Week1	a b	c d	e f g	h	a b c d e f g h i
Week2	j k l	m	n o	p q	a b c d e f g h i j k l m n o p q r s

With this schedule, you need at least nine tapes: one for the level 0, four for the Fridays, and four daily tapes, assuming you reuse daily tapes each week, which is not recommended. If you save the weekly tapes for a month, you need 21 tapes.

If you need to restore the complete file system, you need six tapes: the level 0, the most recent Friday tapes, and *all* the daily tapes for that week.

Recommendations for Choosing a Backup Schedule

For most purposes, the 99995 (“nine-to-five”) schedule is adequate. However, you may want to consider a number of factors in choosing the schedule that is best for your site. For example, do you want to minimize the number of tapes, the time spent doing backups, the time doing a full restore on a damaged file system, or the time spent retrieving individual files that get accidentally deleted? In fact, if you do not need to minimize time and media spent on backups, you could do level 0 backups every day.

Here are some things to consider in choosing your backup schedule:

- You should probably do a level 0 backup of your root file system between once a week to once a month, depending on how much it changes. Back it up daily if `/var` and its mail spool files are in the root slice.
- You should do a full backup on most file systems (that need backups) at least once a month. Keep in mind that frequent level 0 backups produce large backup files that take a long time to write. It also may take a long time to retrieve individual files, since the drive has to move sequentially to the point on the tape where the file is located. Retrieving individual files may be easier if you run incremental backups to pick up the small changes in the file systems you back up. Finding which incremental tape a file is on can take time, however.
- If you have a file system that is used for an application like word processing, it may be very important to you to save a historical record of all the different versions of files across time. Consider doing incremental backups every working day, commonly at level 9. This schedule saves all files modified that day, as well as those files still on disk that were modified since the last backup of a level lower than 9. Remember that a file changed on Tuesday and then again on Thursday, goes onto Friday's lower-level backup looking like it did Thursday night, not Tuesday night. If a user needs the Tuesday version, you cannot restore it unless you have a Tuesday backup tape (or a Wednesday backup tape, for that matter). Similarly, a file present on Tuesday and Wednesday, but removed on Thursday, does not appear on the Friday lower-level backup. If you need the ability to restore different versions of files, be sure you do not reuse the same tape for the daily incremental backups.
- Save a week's worth of daily level 9 backups at least until you do a backup at a level lower than the daily level. However, you should save the daily tapes longer if you want to save different versions of files.
- If you are most concerned about being able to restore quickly a complete file system to its most recent state, do lower-level backups more frequently.
- If you are most concerned about minimizing the number of tapes you use, increase the level of the incremental backups done across the week, so only changes from day to day are saved on each daily tape. In addition, you can increase the level of the backups done at the end of the week, so only

changes from week to week (rather than the whole month) are saved on the weekly tapes. Finally, you can try to put each day's and week's incremental backup onto the same tape by not rewinding the tapes after each use.

- If you are backing up a number of file systems on the same server, you may want to offset the schedule for different file systems, so you are not doing all level 0s on the same day.

Example Backup Strategy for a Server

Table 6-14 shows an example backup strategy for a heavily used file server on a small network where users are doing file-intensive work, such as program development or document production. It assumes that the backup period begins on a Sunday and consists of four seven-day weeks.

Table 6-14 Schedule of Backups for an Example Server

Directory	Date	Level	Tape Name
/	1st Sunday	0	<i>n</i> tapes
/usr	1st Sunday	0	"
/export	1st Sunday	0	"
/export/home	1st Sunday	0	"
	1st Monday	9	A
	1st Tuesday	9	B
	1st Wednesday	5	C
	1st Thursday	9	D
	1st Friday	9	E
	1st Saturday	5	F
/	2nd Sunday	0	<i>n</i> tapes
/usr	2nd Sunday	0	"
/export	2nd Sunday	0	"
/export/home	2nd Sunday	0	"
	2nd Monday	9	G
	2nd Tuesday	9	H
	2nd Wednesday	5	I

Table 6-14 Schedule of Backups for an Example Server (Continued)

Directory	Date	Level	Tape Name
	2nd Thursday	9	J
	2nd Friday	9	K
	2nd Saturday	5	L
/	3rd Sunday	0	<i>n</i> tapes
/usr	3rd Sunday	0	"
/export	3rd Sunday	0	"
/export/home	3rd Sunday	0	"
	3rd Monday	9	M
	3rd Tuesday	9	N
	3rd Wednesday	5	O
	3rd Thursday	9	P
	3rd Friday	9	Q
	3rd Saturday	95	R
/	4th Sunday	0	<i>n</i> tapes
/usr	4th Sunday	0	"
/export	4th Sunday	0	"
/export/home	4th Sunday	0	"
	4th Monday	9	S
	4th Tuesday	9	T
	4th Wednesday	5	U
	4th Thursday	9	V
	4th Friday	9	W
	4th Saturday	5	X

With this plan, you use $4n$ tapes (the number of tapes needed for four full backups of /, /usr, /export, and /export/home) plus 24 additional tapes for the incremental backups of /export/home. This plan assumes that each incremental backup uses one tape and you save the tapes for a month.

Here is how this plan works:

1. On each Sunday, do a full backup (level 0) of `/`, `/usr`, `/export`, and `/export/home`. Save the level 0 tapes for at least 3 months. Depending on your sites needs, you may want to save them longer.
2. On the first Monday of the month, use tape A to do a level 9 backup of `/export/home`. `ufsdump` copies all files changed since the previous lower-level backup, in this case, the level 0 backup that you did on Sunday.
3. On the first Tuesday of the month, use tape B to do a level 9 backup of `/export/home`. Again, `ufsdump` copies all files changed since the last lower-level backup—Sunday’s level 0 backup.
4. On the first Wednesday, use tape C to do a level 5 backup. `ufsdump` copies all file changed since Sunday.
5. Do the Thursday and Friday level 9 backups on tapes D and E. `ufsdump` copies all files changed since the last lower-level backup—Wednesday’s level 5 backup.
6. On the first Saturday of the month, do a level 5 backup of `/export/home`, which copies all files changed since the previous lower-level backup—in this case, the level 0 backup you did on Sunday. Store tapes A-F until the first Monday of the next 4-week period, when you use them again.
7. Repeat steps 1–6 for the next three weeks, using tapes G-L and $4n$ tapes for the level 0 on Sunday, and so on.
8. For each 4-week period, repeat steps 1–7, using a new set of tapes for the level 0s and reusing tapes A–X for the incremental backups. The level 0 tapes could be reused after 3 months.

This plan allows you to save files in their various states for a month. It requires many tapes, but does ensure that you have a library of tapes to draw on in case a user needs to work on an old project that was canceled and then started up again. If you cannot afford so many tapes, you could reuse Tapes A-F each week.

Backing Up Files and File Systems



This chapter contains these sections:

<i>Preparing to Do Backups</i>	<i>page 121</i>
<i>Doing Complete Backups</i>	<i>page 126</i>
<i>Doing Incremental Backups</i>	<i>page 128</i>
<i>Backing Up Individual Files and Directories</i>	<i>page 128</i>
<i>Using a Remote Drive to Do Backups</i>	<i>page 128</i>
<i>Doing Backups on Remote Systems</i>	<i>page 130</i>
<i>Troubleshooting When the Root File System Fills Up</i>	<i>page 130</i>
<i>Options and Arguments for the ufsdump Command</i>	<i>page 131</i>

Preparing to Do Backups

Before you start a backup procedure, you need to know:

- The raw device name for the file systems you want to back up
- The type of tape drive you will use
- The device name for the tape drive and whether the drive is local or remote
- The number of tapes you will need

This section shows you how to find this information.

▼ **How to Find File System Raw Device Names**

1. **Type `more /etc/vfstab` and press Return.**
The contents of the `/etc/vfstab` file are displayed.
2. **Look in the `mount point` column for the name of the file system.**
3. **Use the raw device name in the `device to fsck` column with the `ufsdump` command.**

The example below shows the raw device name to back up the `/usr` file system on venus is `/dev/rdisk/c0t1d0s6`.

```

venus% more /etc/vfstab
#device          device          mount          FS          fsck          auto-          mount
#to mount        to fsck         point          type        pass         mount?       options
#
/proc            -              /proc          proc        -            no           -
swap            -              /tmp           tmpfs       -            yes          -
/dev/dsk/c0t3d0s0 /dev/rdisk/c0t3d0s0 /              ufs         1            no           -
/dev/dsk/c0t3d0s1 -              -              swap        -            no           -
/dev/dsk/c0t1d0s6 /dev/rdisk/c0t1d0s6 /usr           ufs         2            no           -
neptune:/export//usr/openwin -              /usr/openwin   nfs         no           yes          -
neptune:/export/usr/man -              /usr/man       nfs         no           yes          -
venus%

```

▼ **How to Find the Block Device Name for a Mounted File System**

1. **Type `devnm mount-point` and press Return.**
The block device name for the file system is displayed.

```

venus% devnm /usr
/dev/dsk/c0t1d0s6 /usr
venus%

```

2. **Use `/dev/rdisk/device-name` with the `ufsdump` command.**

▼ How to Display the Slices that Contain File Systems

1. Become superuser.

2. Type `prtvtoc /dev/rdisk/device-name` and press Return. Information for all non-zero slices is displayed.

```
venus% su
Password:
# prtvtoc /dev/rdisk/c0t3d0s2
* /dev/rdisk/c0t3d0s2 partition map
*
* Dimensions:
*   512 bytes/sector
*   35 sectors/track
*   6 tracks/cylinder
*   210 sectors/cylinder
*   1019 cylinders
*   974 accessible cylinders
*
* Flags:
*  1: unmountable
* 10: read-only
*
*
* Partition  Tag  Flags      First   Sector   Last
* Partition  Tag  Flags      Sector  Count    Sector  Mount Directory
*   0         2    00         0      32760    32759  /
*   1         3    01      32760    65520    98279
*   2         5    01         0     204540   204539
*   6         4    00      98280   106260   204539
#
```

▼ **How to Determine the Type of a Tape Drive**

You can use the `status` option to the `mt` command to get status information about the Xylogics 472 1/2-inch tape drive and the Exabyte EXB-8200 8-mm tape drive.

The `status` command also reports information about these 1/4-inch tape drives:

- Sysgen (QIC-24)
- Emulex MT-02 (QIC-24)
- Archive QIC-150
- Wangtek QIC-150

To determine what type of tape drive you have:

1. **Load a tape into the drive you want information about.**
2. **Type `mt -f /dev/rmt/0 status` and press Return.**
3. **Repeat the command, substituting tape drive numbers 1, 2, 3, and so on to display information about all available tape drives.**

This example shows status for an Emulex drive (`/dev/rmt/0`) and an Exabyte tape drive (`/dev/rmt/1`) on `venus` and a QIC-150 tape drive on `mars`:

```

venus% mt -f /dev/rmt/0 status
Emulex MT-02 QIC-24 tape drive:
sense key(0x2)= not ready residual= 0 retries= 0
file no= 0 block no= 0
venus% mt -f /dev/rmt/1 status
Exabyte EXB-8200 8mm tape drive:
sense key(0x0)= NO Additional Sense residual= 0  retries= 0
file no= 0  block no= 0
venus% rlogin mars
Password:
mars% mt -f /dev/rmt/0 status
Archive QIC-150 tape drive:
    sense key(0x0)= No Additional Sense  residual= 0  retries= 0
    file no= 0  block no= 0
mars%

```

Here is an quick way to poll a system and locate all tape drives. In this example, the tape drive is at 0:

```

mars% sh
$ for drive in 0 1 2 3 4 5 6 7
> do
> mt -f /dev/rmt/$drive status
> done
Archive QIC-150 tape drive:
  sense key(0x0)= No Additional Sense   residual= 0   retries= 0
  file no= 0   block no= 0
/dev/rmt/1: No such file or directory
/dev/rmt/2: No such file or directory
/dev/rmt/3: No such file or directory
/dev/rmt/4: No such file or directory
/dev/rmt/5: No such file or directory
/dev/rmt/6: No such file or directory
/dev/rmt/7: No such file or directory
$ exit
mars%

```

▼ How to Find Out How Many Tapes You Need

To find out how many tapes you need to do a full backup on a file system:

1. **Type** `ufsdump [options] S backup-device filesystem` **and press Return.**
The `S` option estimates the size in bytes of the incremental backup.
2. **Divide the estimated size by the capacity of the tape to see how many tapes you need.**
See Table 6-3 on page 102 for a list of tape capacities.

Example: How to Find Out How Many Tapes You Need

In this example, the file system of 489472 bytes will fit on one QIC-150 150-Mbyte tape:

```

# ufsdump 0S /dev/rdisk/c0t3d0s7
489472
#

```

Doing Complete Backups

Before you perform the backup procedure, there are several steps you need to take to prepare for the backup.

Note – It's best to perform backups in single-user mode or with the file system unmounted. When the file system is active while it is being backed up, some data may not be included in the backup. If directory level operations (creating, removing, and renaming files and directories) are done while the file system is being backed up, you may not be able to correctly restore data from the backup tape.

▼ How to Do a Complete Backup on Cartridge Tape

This procedure specifies a full level 0 backup on cartridge tape, updating the `/etc/dumpdates` file.

1. Take the system down to single-user level.

- a. On a server, become superuser, change to the root directory, run `shutdown`, then reboot and come up in single-user mode.

```
# cd /
# shutdown
Various messages from shutdown appear
#
# halt
ok boot -s
```

- b. On a standalone system, become superuser, then type `init s` and press Return.

2. [Optional] If you think the file system has problems, type `fsck raw_device` and press Return.

3. Insert a tape that is not write protected into the tape drive.

4. Type `/usr/bin/ufsdump 0ucf /dev/rmt/unit /dev/rdisk/cntndnsn` and press Return.

The `0` option specifies a complete level 0 backup, the `u` option updates the `/etc/dumpdates` file; the `c` option specifies a cartridge tape drive; and the `f` option indicates that you specify the tape drive file name as part of the command-line argument. The tape is rewound unless you use the `n` option after the tape unit number (for example, `/dev/rmt/0n`). See “Options and Arguments for the `ufsdump` Command” on page 131 for a complete description of the options.

5. When prompted, remove the tape and replace with the next volume.

6. Label each tape with the volume number, level, date, system name, and file system.

Examples: Full Backups

To make a full dump of a root file system on `c0t3d0s0`, on a 150-Mbyte cartridge tape (`/dev/rmt/0`):

```
# ufsdump 0cf dev/rmt/0 /dev/rdsk/c0t3d0s0
```

In this example, a full backup is made of `/dev/dsk/c0t1d0s0`.

```
moon:# ufsdump 0uf /dev/rmt/0 /dev/dsk/c0t1d0s0
DUMP: Writing 32 Kilobyte records
DUMP: Date of this level 0 dump: Thu Feb 10 17:28:54 1994
DUMP: Date of last level 0 dump: the epoch
DUMP: Dumping /dev/rdsk/c0t1d0s0 to /dev/rmt/0.
DUMP: Mapping (Pass I) [regular files]
DUMP: Mapping (Pass II) [directories]
DUMP: Estimated 42934 blocks (20.96MB).
DUMP: Dumping (Pass III) [directories]
DUMP: Dumping (Pass IV) [regular files]
DUMP: Tape rewinding
DUMP: 42878 blocks (20.94MB) on 1 volume at 603 KB/sec
DUMP: DUMP IS DONE
DUMP: Level 0 dump on Thu Feb 10 17:28:54 1994
moon:#
```

Doing Incremental Backups

Plan your backup schedule using information from “Planning a Backup Schedule” on page 112. This section provides instructions for how to do incremental backups.

▼ How to Back Up Incremental Changes

- 1. Bring the system to single-user mode.**
- 2. Become superuser.**
- 3. Put a tape into the tape drive.**
- 4. Type `ufsdump [1-9]ucf /dev/rmt/unit /dev/rdisk/cntndnsn` and press Return.**
Type the level of the backup at the beginning of the `ufsdump` arguments. For example, to do a level 9 backup, type `9ucf`.
- 5. Remove the tape from the tape drive and label it.**

Backing Up Individual Files and Directories

You use the `ufsdump` command to back up individual files and directories.

▼ How to Back Up Individual Files and Directories

- ◆ **Type `ufsdump [options] tape-drive filenames`**
Where *filenames* is one or more individual file or directory names separated by spaces, for example, `/home/user1/mail /home/user2/mail`.

Using a Remote Drive to Do Backups

You can use the `ufsdump` command to back up files from one system to a drive on another system. The command syntax is:

```
ufsdump [options] remote-host:dump-file files-to-backup
```

Note – The naming convention you use for the remote drive depends on the system where the drive resides. Use the naming conventions that match the SunOS release on the system with the remote tape drive.

To run a backup as a different user on a remote host, the command syntax is as follows:

```
ufsdump [options] user@remote-host:dump-file files-to-backup
```

The only difference between this command and one you use to back up to a local drive is that you preface the *dump-file* argument with the *remote-host* name followed by a colon (:).

▼ How to Find Out if You Can Access a Remote Drive

1. Become superuser.

2. Type `rsh remote-host cat /etc/motd` and press Return.

If a message like this is displayed, your server is in the remote system's `/.rhosts` file.

```
neptune% rsh mars cat /etc/motd
Sun Microsystems, Inc. SunOS 5.4    May 1994
neptune%
```

If the message `Unknown host` is displayed, contact the person responsible for the remote system to add the local system to the `/.rhosts` file. If you are not using Name Server, NIS, or NIS+, you must also add the IP address for the remote machine to the local `/etc/hosts` file.

Doing Backups on Remote Systems

You can do backups on remote systems by remote login (`rlogin`) to the system and typing the backup command. If the tape drive is local, use the usual command line syntax. If the tape drive is on a remote system, specify the server name as part of the command-line argument.

▼ How to Back Up SunOS 5.x Systems on a Remote System

1. Add the system name to the server's `.rhosts` file and the IP address to the server's `/etc/hosts` file, if necessary.
2. Type `rlogin host-name` and press Return.
3. Type `ufsdump [options] server-name:/dev/rmt/unit files-to-back-up`.

▼ How to Back Up SunOS 5.x Systems to a Remote SunOS 4.x System

1. Type `rlogin host-name` and press Return.
2. Type `ufsdump [options] server-name:/dev/rst/unit files-to-back-up`.
Note that you use the old-style tape drive device name if the tape drive is on a SunOS 4.x system.

▼ How to Back Up SunOS 4.x Systems to a Remote Sun OS 5.x System

1. Type `rlogin host-name` and press Return.
2. Type `dump [options] server-name:/dev/rmt/unit files-to-back-up`.
Notice that you run the `dump` command on the SunOS 4.x system.

Troubleshooting When the Root File System Fills Up

Symptom:

You do a backup of a file system. Nothing is written to the media, but the root file system fills up. The `ufsdump` command prompts you to install the second volume of media when the root file system is full. `Filesystem is full` messages will be displayed in the console window.

Explanation:

If you used an invalid destination device name with the `f` option, the `ufsdump` command wrote to a file in the `/dev` directory of the root file system, filling it up. For example, if you typed `/dev/rmt/st0` instead of `/dev/rmt/0`, the backup file `/dev/rmt/st0` was created on the disk rather than being sent to the tape drive.

Resolution:

1. Type `cd /dev/rmt` and press Return.
2. Type `ls -l` and press Return.
A list of tape devices is displayed.
3. Look at the file size for any unusually large files or files that are not device special (no major or minor device numbers) or symbolic links.
4. Become superuser.
5. Type `rm filename` and press Return.

This example shows the contents of a typical `/dev/rmt` directory, with no unusually large files:

```
venus% cd /dev/rmt
venus% ls -l
total 0
crw-rw-rw- 1 root      18,  8 Sep 12 20:45 0
crw-rw-rw- 1 root      18,  8 Sep 12 20:45 0h
crw-rw-rw- 1 root      18, 12 Sep 12 20:45 0hn
crw-rw-rw- 1 root      18,  0 Sep 12 20:45 0l
crw-rw-rw- 1 root      18,  4 Sep 12 20:45 0ln
crw-rw-rw- 1 root      18,  8 Sep 12 20:45 0m
crw-rw-rw- 1 root      18, 12 Sep 12 20:45 0mn
crw-rw-rw- 1 root      18, 12 Sep 12 20:45 0n
venus%
```

Options and Arguments for the `ufsdump` Command

This section describes in detail the options and arguments for the `ufsdump` command. The syntax for the `ufsdump` command is:

`/usr/sbin/ufsdump [options] [arguments] files-to-back-up`

options is a single string of one-letter option names.

arguments may be multiple strings.

The option letters and the arguments that go with them must be in the same order and the *files-to-back-up* argument must come last.

Default Command Options

If you run the `ufsdump` command without any options, using this syntax:

`ufsdump files-to-back-up`

`ufsdump` uses these options, by default:

`ufsdump 9uf /dev/rmt/0 files-to-back-up`

These options do a level 9 incremental backup to the default tape drive at its preferred density.

Options for the `ufsdump` Command

Table 7-1 describes the options for the `ufsdump` command.

Table 7-1 Options for the `ufsdump` Command

Option	Description
0-9	Backup level. Level 0 is for a full backup of the whole file system specified by <i>files-to-backup</i> . Levels 1-9 are for incremental backups of files that have changed since the last lower-level backup.
a <i>archive-file</i>	Archive file. Store (archive) a backup table of contents in a specified file on the disk. The file can be understood only by <code>ufsrestore</code> , which uses it to determine whether a file to be restored is present in a backup file, and if so, on which volume of the media it resides.
b <i>factor</i>	Blocking factor. The number of 512-byte blocks to write to tape per operation.
c	Cartridge. Back up to cartridge tape. When end-of-media detection applies, this option sets the block size to 126.

Table 7-1 Options for the `ufsdump` Command (Continued)

Option	Description
<code>d bpi</code>	Tape density. You need to use this option only when <code>ufsdump</code> cannot detect the end of the media.
<code>D</code>	Diskette. Back up to diskette.
<code>f dump-file</code>	Dump file. Write the files to the destination specified by <i>dump-file</i> instead of the default device. If the file is specified as <i>user@system: device</i> , <code>ufsdump</code> attempts to execute as the specified user on the remote system. The specified user must have a <code>.rhosts</code> file on the remote system that allows the user invoking the command on the local system to access the remote system.
<code>l</code>	Autoload. Use this option if you have an autoloading (stackloader) tape drive. When the end of a tape is reached, this option takes the drive offline and waits up to two minutes for the tape drive to be ready again. If the drive is ready within two minutes, it continues. If it is not ready after two minutes, it prompts an operator to load another tape.
<code>n</code>	Notify. When intervention is needed, send a message to all terminals of all users in the <code>sys</code> group.
<code>o</code>	Offline. When finished with a tape or diskette, take the drive offline, rewind (if tape), and if possible remove the media (for example, eject a diskette or remove 8-mm autoloading tape).
<code>s size</code>	Size. Specify the length of tapes in feet or number of 1024-byte blocks for diskettes. You need to use this option only when <code>ufsdump</code> cannot detect the end of the media.
<code>S</code>	Estimate size of backup. Determine the amount of space that is needed to perform the backup, without actually doing it, and output a single number indicating the estimated size of the backup in bytes.
<code>t tracks</code>	Tracks. Specify the number of tracks for 1/4-inch cartridge tape. You need to use this option only when <code>ufsdump</code> cannot detect the end of the media. See “End-of-Media Detection” on page 137 for more information.
<code>u</code>	Update the dump record. For a completed backup on a file system, add an entry to the file <code>/etc/dumpdates</code> . The entry indicates the device name for the file system’s disk slice, the backup level (0–9), and the date. No record is written when you do not use the <code>u</code> option or when you back up individual files or directories. If a record already exists for a backup at the same level, it is replaced.

Table 7-1 Options for the `ufsdump` Command (Continued)

Option	Description
v	Verify. After each tape or diskette is written, verify the contents of the media against the source file system. If any discrepancies occur, prompt the operator to mount new media, then repeat the process. Use this option on an unmounted file system only, because any activity in the file system causes it to report discrepancies.
w	Warning. List the file systems appearing in <code>/etc/dumpdates</code> that have not been backed up within a day. When you use this option all other options are ignored.
W	Warning with highlight. Show all the file systems that appear in <code>/etc/dumpdates</code> and highlight those file systems that have not been backed up within a day. When you use this option all other options are ignored.

Note – The `/etc/vfstab` file does not contain information about how often to back up a file system.

Backup Device (*dump-file*) Argument

The *dump-file* argument (to the `f` option) specifies the destination of the backup, which can be one of the following:

- Local tape drive or diskette drive
- Remote tape drive or diskette drive
- Standard output

Use this argument when the destination is not the default local tape drive `/dev/rmt/0`. If you use the `f` option, then you must specify a value for *dump-file*.

Note – The *dump-file* argument can also point to a file on a local or remote disk, which, if used by mistake, can fill up a file system.

Local Tape or Diskette Drive

Typically, *dump-file* specifies a raw device file for a tape or diskette drive. When `ufsdump` writes to an output device, it creates a single backup file which may span multiple tapes or diskettes.

You specify the tape or diskette device on your system using a device abbreviation. The first device is always 0. For example, if you have a SCSI tape controller and one QIC-24 tape drive that uses medium-density formatting, use this device name:

```
/dev/rmt/0m
```

When you specify a tape device name, you can also type the letter “n” at the end of the name to indicate that the tape drive should not rewind after the backup is completed. For example:

```
/dev/rmt/0mn
```

Use the “no-rewind” option if you want to put more than one file onto the tape. If you run out of space during a backup, the tape does not rewind before `ufsdump` asks for a new tape. See “Backup Device Names” on page 103 for a complete description of device naming conventions.

Remote Tape or Diskette Drive

You specify a remote tape or diskette drive using the syntax *host:device*. `ufsdump` writes to the remote device when `root` on the local system has access to the remote system. If you usually run `ufsdump` as superuser, the name of the local system must be included in the `.rhosts` file of the remote system. If you specify the device as *user@host:device*, `ufsdump` tries to execute as the specified user on the remote system. In this case, the specified user must have a `.rhosts` file on the remote system to allow the user to access the remote system.

Use the naming convention for the device that matches the operating system for the system on which the device resides, not the system from which you run the `ufsdump` command. If the drive is on a system that is running a previous SunOS release (for example, 4.1.1), use the SunOS 4.x device name (for example, `/dev/rst0`). If the system is running SunOS 5.x system software, use the SunOS 5.x convention (for example, `/dev/rmt/0m`).

Note – You must specify remote devices explicitly with the *dump-file* argument. In previous SunOS releases `rdump` directed the output to the remote device defined by the `dumphost` alias. `ufsdump` does not have an `rufsdump` counterpart.

Standard Output

When you specify a dash (-) as the *dump-file* argument, `ufsdump` writes to the standard output.

Note – The `v` option (verify) does not work when the *dump-file* argument is standard output.

You can use the `ufsdump` and `ufsrestore` commands in a pipeline to copy a file system by writing to the standard output with `ufsdump` and reading from the standard input with `ufsrestore`, as shown in this example:

```
# ufsdump 0f - /dev/rdisk/c0t0d0s7 | (cd /home; ufsrestore xf -)
```

Specifying Files to Backup

You must always include *files-to-backup* as the last argument on the command line. This argument specifies the source or contents of the backup. It usually identifies a file system but can also identify individual files or directories.

For a file system, specify the raw device file for a disk slice. It includes the disk controller abbreviation (*c*), the target number (*t*) for SCSI devices only, a number indicating the disk number (*d*), and the slice number (*s*). For example, if you have a SCSI disk controller on your standalone system (or server) and you want to back up `/usr` located in slice 6, specify the device as follows:

```
/dev/rdisk/c0t0d0s6
```

You can specify the file system by its mount point directory (for example, `/home`), as long as there is an entry for it in the `/etc/vfstab` file.

See “Backup Device Names” on page 103 for a complete description of device naming conventions.

For individual files or directories, type one or more names separated by spaces.

Note – When you use `ufsdump` to back up one or more directories (rather than a whole file system) a level 0 backup is done. Incremental backups do not apply.

End-of-Media Detection

`ufsdump` automatically detects the end-of-media for most devices. Therefore, you do not usually need to use the `c`, `d`, `s`, and `t` options to perform multivolume backups.

The only time you need to use the end-of-media options is when `ufsdump` does not understand the way the device detects the end-of-media or you are going to restore the files on a system with an older version of the `restore` command. To ensure compatibility with older versions of the `restore` command, the `size` option can still force `ufsdump` to go to the next tape or diskette before reaching the end of the current tape or diskette.

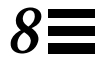
Specifying Tape Characteristics

If you do not specify any tape characteristics, the `ufsdump` command uses a set of defaults. Table 7-2 shows some arguments to the `ufsdump` command that work well for different types of tape cartridges. You can specify tape cartridge (`c`), density (`d`), size (`s`), and number of tracks (`t`). Note that you can specify the options in any order as long as the arguments that follow match the order of the options.

Table 7-2 Arguments to `ufsdump` to Specify Tape Capacity

Tape	Arguments
Diskette	<code>ufsdump Ds 1422</code>
60-Mbyte cartridge	<code>ufsdump cdst 1000 425 9</code>
150-Mbyte cartridge	<code>ufsdump cdst 1000 700 18</code>
1/2-inch tape	<code>ufsdump dsb 1600 2300 126</code>
2.3-Gbyte 8-mm tape	<code>ufsdump dsb 54000 6000 126</code>
5.0-Gbyte 8-mm tape	<code>ufsdump dsb 54000 13000 126</code>
5.0-Gbyte 4-mm tape	<code>ufsdump b 96</code>

Restoring Files and File Systems



This chapter has these sections:

<i>Preparing to Restore Files and File Systems</i>	<i>page 139</i>
<i>Restoring Complete File Systems</i>	<i>page 142</i>
<i>Restoring Individual Files and Directories</i>	<i>page 146</i>
<i>Using a Remote Drive to Restore Files</i>	<i>page 150</i>
<i>Troubleshooting</i>	<i>page 150</i>
<i>Options and Arguments for the ufsrestore Command</i>	<i>page 151</i>

This chapter describes how to use the `ufsrestore(1M)` command to restore files and file systems that were backed up using the `ufsdump` command. See Chapter 4, “Copying UFS Files and File Systems,” for information about other commands you can use to archive, restore, copy, or move files and file systems.

Preparing to Restore Files and File Systems

The `ufsrestore` command copies files from backups created using the `ufsdump` command into the current working directory. You can use `ufsrestore` to reload an entire file system hierarchy from a level 0 dump and incremental dumps that follow it or to restore one or more single files from any dump tape. If `ufsrestore` is run by root, files are restored with their original owner, last modification time, and mode (permissions).

Before you start to restore files or file systems, you need to know:

- Which tapes (or diskettes) you need
- The raw device name for the file systems you want to back up
- The type of tape drive you will use
- The device name (local or remote) for the tape drive

This section describes how to find this information.

Determining Which Tapes to Use

Before you can begin restoring file systems or files, you must determine which backup tapes you need.

▼ How to Determine Which Tapes to Use for a Complete or Incremental Restore

When restoring an entire file system, you always need the most recent level 0 backup tape. You also need the most recent incremental backup tapes made at each of the higher levels. Refer to the backup plan that you are using to determine the levels and number of tapes you need. See “Planning a Backup Schedule” on page 112 for more information.

For example, if you make level 0 and level 9 backups, you need the most recent level 0 backup tape and the last level 9 backup tape made.

▼ How to Determine Which Tapes to Use to Restore Individual Files or File Systems

1. Ask the user the date when the file or file system was lost, or the approximate date of the files to be recovered.

2. Refer to your backup plan to find the date of the last backup that would have the file or file system on it.

Note that you do not necessarily use the most recently backed up version of the file. To retrieve the most recent version of a file, work backward through the incremental backups from highest to lowest level and most recent to least recent.

- 3. If you have online archive files, type `ufsrestore ta archive-name ./path/filename(s)` and press Return.**
Be sure to use the complete path for the *filename(s)*. A list of the files and the media they are stored on is displayed.
- 4. Retrieve the media containing the backups.**
Be aware of the storage organization of backup media at your site, so that you can locate media that are months or years old.
- 5. [Optional] Insert media in the drive and type.**
`ufsrestore tf device-name ./path/filename(s)` and press Return.
Be sure to use the complete path for the *filename(s)*. If a file is in the backup, its name and inode number is listed. Otherwise, a message says it is not on the volume.
- 6. If you have multiple dump files on the same tape, you can use the `-s n` option to position the tape at the dump you want to use.**
For example, type `ufsrestore xfs /dev/rmt0 5` and press Return to position the tape at the fifth dump and restore it.

Example: How to List Files from an OnLine Archive or Backup Tape

If you use `ufsdump` to dump the `/usr` slice, the table of contents lists only the files and directories under `/usr`. To see if `/usr/bin/pwd` is in the online archive, type:

```
# ufsrestore ta archive-name ./bin/pwd
```

To see if `/usr/bin/pwd` is on the backup tape, type:

```
# ufsrestore tf /dev/rmt0 ./bin/pwd
```

Determining the Disk Device Name

If you have properly labeled your backup tapes, you should be able to use the disk device name (`/dev/rdisk/cntndnsn`) from the tape label. See “How to Find File System Raw Device Names” on page 122 for more information.

Determining the Type of Tape Drive You Will Use

You must use a tape drive that is compatible with the backup media to restore the files. The format of the backup media determines which drive you must use to restore files. For example, if your backup media is 8-mm tape, you must use an 8-mm tape drive to restore the files.

Determining the Tape Device Name

You may have specified the tape device name (`/dev/rmt/unit`) as part of the backup tape label information. If you are using the same drive to restore a backup tape, you can use the device name from the label. See “Choosing Which Media to Use” on page 102 for more information on media devices and device names.

Restoring Complete File Systems

Occasionally, a file system becomes so damaged that you must completely restore it. Typically, you need to restore a complete file system because of a disk head crash. You may need to replace the hardware before you can restore the software. See *Peripherals Administration* for information on how to replace a disk. Fully restoring a file system such as `/export/home` can take a lot of time. If you have faithfully backed up file systems, you can restore them to their state as of the last incremental backup.

Synopsis of Steps

This section provides an overview of a good strategy for restoring complete file systems:

1. Replace any broken hardware.
2. If replacing a disk:
 - a. Add the defect list and be sure the defect list is found.
 - b. Format the disk.
 - c. Re-create the slices to match the old slices.
 - d. Label the disk.

- e. Remake the file systems.
 - f. Check that the file system was properly remade.
3. Mount the file system on a temporary mount point.
 4. Restore the contents of the latest full backup, and then restore subsequent incremental backups from lowest to highest level.
 5. Unmount the file system from its temporary mount point.
 6. Check the file system for inconsistencies.
 7. Mount the file system at its permanent mount point.

▼ **How to Restore a Complete File System (not / or /usr)**

1. **Become superuser.**
2. **If necessary, type `umount /dev/rdisk/cntndnsn` and press Return.**
3. **Type `newfs /dev/rdisk/cntndnsn` and press Return.**
You are asked if you want to construct a new file system on the device.
4. **Type `y` and press Return.**
The new file system is created.
5. **Type `mount /dev/dsk/cntndnsn /mnt` and press Return.**
The file system is mounted on a temporary mount point.

Note – You specify the block device directory (`/dev/dsk`) to mount the disk, not the raw device directory.

6. **Type `cd /mnt` and press Return.**
You have changed to the mount-point directory.
7. **Write-protect the tapes for safety.**
8. **Insert the first volume of the level 0 tape in the tape drive.**
9. **Type `ufsrestore rvf /dev/rmt/unit` and press Return.**
The level 0 dump is restored. If the dump required multiple tapes, you will be prompted to load the next tape.

10. **Remove the tape and load the next tape in the drive.**
Always restore tapes starting with 0 and continuing until you reach the highest level.
11. **Type `ufsrestore rvf /dev/rmt/unit` and press Return.**
The next level tape is restored. If the dump required multiple tapes, you will be prompted to load the next tape.
12. **Type `rm restoresymtable` and press Return.**
The `restoresymtable` created by `ufsrestore` is removed.
13. **Type `cd /` and press Return.**
14. **Type `umount /mnt` and press Return.**
The file system is unmounted.
15. **Remove the last tape and insert a new tape that is not write-protected in the tape drive.**
16. **Type `ufsdump 0uf /dev/rmt/unit /dev/rdisk/cntndnsn` and press Return.**
You should always do an immediate backup of a newly created file system because `ufsrestore` repositions the files and changes the inode allocation.
17. **Type `mount /dev/dsk/cntndns` and press Return.**
The restored file system is mounted and available for use.

Example: How to Restore a Complete File System

In this example, the file system on `/dev/rdisk/c0t3d0s7` is restored:

Code Example 8-1 Restoring /dev/rdisk/c0t3d0s7 (1 of 3)

```
# newfs /dev/rdisk/c0t3d0s7
newfs: construct a new file system /dev/rdisk/c0t3d0s7: (y/n)? y
/dev/rdisk/c0t3d0s7 406296 sectors in 1254 cylinders of 9 tracks, 36 sectors
                208.0MB in 79 cyl groups (16 c/g, 2.65MB/g, 1216 i/g)
super-block backups (for fsck -F ufs -o b=#) at:
 32, 5264, 10496, 15728, 20960, 26192, 31424, 36656, 41888,
47120, 52352, 57584, 62816, 68048, 73280, 78512, 82976, 88208,
93440, 98672, 103904, 109136, 114368, 119600, 124832, 130064, 135296,
140528, 145760, 150992, 156224, 161456, 165920, 171152, 176384, 181616,
186848, 192080, 197312, 202544, 207776, 213008, 218240, 223472, 228704,
```

Code Example 8-1 Restoring /dev/rdisk/c0t3d0s7 (2 of 3)

```
233936, 239168, 244400, 248864, 254096, 259328, 264560, 269792, 275024,
280256, 285488, 290720, 295952, 301184, 306416, 311648, 316880, 322112,
327344, 331808, 337040, 342272, 347504, 352736, 357968, 363200, 368432,
373664, 378896, 384128, 389360, 394592, 399824, 405056,
# mount /dev/dsk/c0t3d0s7 /mnt
# cd /mnt
# ufsrestore rvf /dev/rmt/0
Verify volume and initialize maps
Media block size is 126
Dump   date: Wed Mar 11 10:16:53 1994
Dumped from: the epoch
Level 0 dump of /export/home on mars:/dev/dsk/c0t3d0s7
Label: none
Begin level 0 restore
Initialize symbol table.
Extract directories from tape
Calculate extraction list.
Warning: ./lost+found: File exists
Make node ./pubs
Make node ./pubs/.wastebasket
Make node ./pubs/Junk
Extract new leaves.
Check pointing the restore
extract file ./pubs/.cshrc
extract file ./pubs/.login
extract file ./pubs/.profile
extract file ./pubs/.Xdefaults
extract file ./pubs/.openwin-init.BAK
extract file ./pubs/junk
extract file ./pubs/.desksetdefaults
extract file ./pubs/dead.letter
extract file ./pubs/.openwin-init
extract file ./pubs/.mtdeletelog
extract file ./pubs/backup.examples
extract file ./pubs/core
extract file ./pubs/backup.examples%
extract file ./pubs/.Xauthority
Add links
Set directory mode, owner, and times.
Check the symbol table.
Check pointing the restore
# ls
```

Code Example 8-1 Restoring /dev/rdisk/c0t3d0s7 (3 of 3)

```

./                lost+found/      restoresymtable
../              pubs/
# rm restoresymtable
# cd /
# umount /mnt
# ufsdump 0ucf /dev/rmt/0 /dev/rdisk/c0t3d0s7
DUMP: Writing 63 Kilobyte records
DUMP: Date of this level 0 dump: Wed Mar 11 10:30:58 1994
DUMP: Date of last level 0 dump: the epoch
DUMP: Dumping /dev/rdisk/c0t3d0s7 (/export/home) to /dev/rmt/0
DUMP: mapping (Pass I) [regular files]
DUMP: mapping (Pass II) [directories]
DUMP: estimated 956 blocks (478KB)
DUMP: dumping (Pass III) [directories]
DUMP: dumping (Pass IV) [regular files]
DUMP: Tape rewinding
DUMP: 956 blocks (478KB) on 1 volume
DUMP: DUMP IS DONE
DUMP: level 0 dump on Wed Mar 11 10:30:58 1994
# mount /dev/dsk/c0t3d0s7
# mount
/ on /dev/dsk/c0t0d0s0 read/write on Wed Mar 11 09:38:31 1994
/usr on /dev/dsk/c0t0d0s6 read/write on Wed Mar 11 09:38:31 1994
/proc on /proc read/write on Wed Mar 11 09:38:31 1994
/dev/fd on fd read/write on Wed Mar 11 09:38:31 1994
/tmp on swap on Wed Mar 11 09:38:33 1994
/opt on /dev/dsk/c0t0d0s7 setuid on Wed Mar 11 09:38:36 1994
/export/home on /dev/dsk/c0t3d0s7 setuid on Wed Mar 11 10:32:02 1994
#

```

Restoring / and /usr File Systems

See the *Administration Supplement for Solaris Platforms* for instructions on how to restore the / and /usr file systems.

Restoring Individual Files and Directories

This section describes how to restore individual files and directories using a local tape drive. See “Using a Remote Drive to Restore Files” on page 150 for information on how to use a remote drive to restore files.

When you restore files in a directory other than the root directory of the file system, `ufsrestore` re-creates the file hierarchy in the current directory. For example, if you restore files to `/home1` that were backed up from `/home/doc/books` the files are restored in the directory `/home1/doc/books`.

When restoring individual files and directories, it is a good idea to restore them to a temporary directory such as `/var/tmp`. After you verify them, you can move the files to their proper locations. You can restore individual files and directories to their original locations. If you do so, be sure you are not overwriting newer files with older versions from the backup tape.

▼ How to Restore Files Interactively

1. **Become superuser.**
2. **Write-protect the tape for safety.**
3. **Put the backup tape in the tape drive.**
4. **Type `cd /var/tmp` and press Return.**
If you want to restore the files to a different directory, substitute the directory name for `/var/tmp` in this step.
5. **Type `ufsrestore if /dev/rmt/unit` and press Return.**
Some informational messages and the `restore >` prompt are displayed.
6. **Create a list of files to be restored.**
 - a. **To list the contents of a directory, type `ls` and press Return.**
 - b. **To change directories, type `cd directory-name` and press Return.**
 - c. **To add a directory or file name to the list of files to be restored, type `add filename` and press Return.**
 - d. **To remove a directory or file name from the list of files to be restored, type `delete filename` and press Return.**
 - e. **To keep the mode of the current directory unchanged, type `setmodes` and press Return. Then type `n` and press Return.**
See “Options and Arguments for the `ufsrestore` Command” on page 151 for more information about the `setmodes` command.
7. **When the list is complete, type `extract` and press Return.**
`ufsrestore` asks you which volume number to use.

8. **Type the volume number and press Return. If you have only one volume, type 1 and press Return.**
The files and directories in the list are extracted and restored to the current working directory.
9. **Type quit and press Return.**
The shell prompt is displayed.
10. **Use the `ls -l` command to list the restored files and directories.**
A list of files and directories is displayed.
11. **Check the list to be sure all the files and directories you specified in the list have been restored.**
12. **Use the `mv` command to move the files to the proper directories.**

Example: How to Restore Files Interactively

In this example, the files `backup.examples` and `junk` are restored from the `pubs` directory:

```
# cd /var/tmp
# ufsrestore if /dev/rmt/0
ufsrestore > ls
.:
lost+found/  pubs/

ufsrestore > cd pubs
ufsrestore > ls
./pubs:
.Xauthority      .login           .profile
backup.examples%
.Xdefaults       .mtdeletelog    .wastebasket/    core
.cshrc           .openwin-init   Junk/             dead.letter
.desksetdefaults .openwin-init.BAK backup.examples  junk

ufsrestore > add backup.examples
ufsrestore > add junk
ufsrestore > setmodes
set owner/mode for '.'? [yn] n
ufsrestore > extract
You have not read any volumes yet.
Unless you know which volume your file(s) are on you should start
```

```
with the last volume and work towards the first.
Specify next volume #: 1
set owner/mode for '.'? [yn] n
ufsrestore > quit
# ls -l
total 6
drwxrwxrwt   3 sys      sys      512 Mar 11 10:36 ./
drwxrwxr-x  18 root     sys      512 Mar 10 16:43 ../
drwxr-xr-x   2 pubs    staff    512 Mar 11 10:11 pubs/
# pwd
/var/tmp
# cd pubs
# ls
./                ../                backup.examples  junk
#
```

▼ How to Restore Specific Files

1. **Become superuser.**
2. **Write-protect the tape for safety.**
3. **Put the backup tape in the tape drive.**
4. **Type `cd /var/tmp` and press Return.**
If you want to restore the files to a different directory, substitute the directory name for `/var/tmp` in this step.
5. **Type `ufsrestore xf /dev/rmt/unit filename` and press Return.**
The `x` option tells `ufsrestore` to copy specific files or directories in the *filename* argument.
6. **Type the volume number where files are located and press Return.**
The file is restored to the current working directory. The message `set owner/mode for '.'? [yn]` is displayed.
7. **Type `n` and press Return.**
Directory modes remain unchanged.
8. **Type `ls -l filename` and press Return.**
A listing for the file is displayed.
9. **Use the `mv` command to move the file to the proper directory.**

Example: How to Restore a Specific File

In this example, the `/pubs/backup.examples` file is restored to the `/var/tmp` directory:

```
# cd /var/tmp
# ufsrestore xf /dev/rmt/0 pubs/backup.examples
You have not read any volumes yet.
Unless you know which volume your file(s) are on you should start
with the last volume and work towards the first.
Specify next volume #: 1
set owner/mode for '.'? [yn] n
# ls
./      ../      pubs/
# cd pubs
# ls
./      ../      backup.examples
#
```

Using a Remote Drive to Restore Files

You can restore files from a remote drive by adding *remote-host:* to the front of the tape device name. Here is the syntax:

```
ufsrestore rf [user@]remote-host:/dev/rmt/unit filename
```

For example, to access a remote tape drive `/dev/rmt/0` on the system `venus`, type:

```
# ufsrestore rf venus:/dev/rmt/0 filename
```

Troubleshooting

Make Sure the Backup and Restore Commands Match

You can only use `ufsrestore` to restore files backed up with `ufsdump`. If you backup with `tar`, restore with `tar`. If you use the `ufsrestore` command to restore a tape that was written with another command, an error message tells you that the tape is not in `ufsdump` format.

Check to Make Sure You Have the Right Current Directory

It is easy to restore files to the wrong location. Because the `ufsdump` command always copies files with full path names relative to the root of the file system, you should usually change to the root directory of the file system before running `ufsrestore`. If you change to a lower-level directory, after you restore the files you will see a complete file tree created under that directory.

Use the Old `restore` Command to Restore Multivolume Diskette Backups

You cannot use the `ufsrestore` command to restore files from a multivolume backup set of diskettes made with the `dump` command. You must restore the files on a SunOS 4.x system.

Options and Arguments for the `ufsrestore` Command

Command Syntax

The syntax of `ufsrestore` is:

```
ufsrestore [options][arguments][filename(s) ...]
```

options is a single string of one-letter option names. *arguments* follows the option string with the arguments that match the options. The option names and the arguments that go with them must be in the same order. The *filename(s)* argument, which goes with either the `x` or `t` options, must always come last.

Note – You must choose one and *only* one of these options: `i`, `r`, `R`, `t`, or `x`.

Options and Arguments

You must use one (and only one) of the `ufsrestore` options shown in Table 8-1.

Table 8-1 One Required Option for the `ufsrestore` Command

Option	Description
<code>i</code>	Interactive. Runs <code>ufsrestore</code> in an interactive mode. In this mode, you can use a limited set of shell commands to browse the contents of the media and select individual files or directories to restore. See “Interactive Commands” on page 154 for a list of available commands.
<code>r</code>	Recursive. Restores the entire contents of the media into the current working directory (which should be the top level of the file system). Information used to restore incremental dumps on top of the full dump (e.g., <code>restoresymtable</code>) is also included. To completely restore a file system, use this option to restore the full (level 0) dump and then for each incremental dump. Although intended for a new file system (one just created with the <code>newfs</code> command), the file system may contain files and files not on the backup media are preserved.
<code>R</code>	Resume restoring. Prompts for the volume from which to resume restoring and restarts from a checkpoint. You rerun the <code>ufsrestore</code> command with this option after a full restore (<code>r</code> option) is interrupted.
<code>x</code> [<i>filename(s)...</i>]	Extract. Selectively restores the files you specify by the <i>filename(s)</i> argument. <i>filename(s)</i> can be a list of files and directories. All files under a specified directory are restored unless you also use the <code>h</code> option also. If you omit <i>filename(s)</i> or enter “.” for the root directory, all files on all volumes of the media (or from standard input) are restored. Existing files are overwritten, and warnings are displayed.
<code>t</code> [<i>filename(s)...</i>]	Table of contents. Checks the files specified in the <i>filename(s)</i> argument against the media. For each file, lists the full file name and the inode number (if the file is found) or indicates the file is not on the “volume” (meaning any volume in a multivolume dump). If you do not enter the <i>filename(s)</i> argument, all files on all volumes of the media are listed (without distinguishing on which volume files are located). If you also use the <code>h</code> option, only the directory files specified in <i>filename(s)</i> , not their contents, are checked and listed. The table of contents is read from the first volume of the media, or, if you use the <code>a</code> option, from the specified archive file. This option is mutually exclusive with the <code>x</code> and <code>r</code> options.

In addition to one of the options shown in Table 8-1, you can choose from the options shown in Table 8-2.

Table 8-2 Additional Options for the `ufsrestore` Command

Option	Description
<code>a</code> <i>archive-file</i> [<i>filename(s)...</i>]	Takes the dump table of contents from the specified <i>archive-file</i> instead of from the media (first volume). You can use this option in combination with the <code>t</code> , <code>i</code> , or <code>x</code> options to check for the files in the dump without having to mount any media. If you use it with the <code>x</code> and interactive extract options, you will be prompted to mount the appropriate volume before extracting the file(s).
<code>b</code> <i>factor</i>	Blocking factor. Number of 512-byte blocks to write to tape per operation. By default, <code>ufsrestore</code> tries to figure out the block size used in writing the tape.
<code>d</code>	Debug. Turn on debugging messages.
<code>f</code> <i>backup-file</i>	Backup file. Reads the files from the source indicated by <i>backup-file</i> , instead of from the default device file <code>/dev/rmt/0m</code> . If you use the <code>f</code> option, you must specify a value for <i>backup-file</i> . When <i>backup-file</i> is of the form <i>system:device</i> , <code>ufsrestore</code> reads from the remote device. You can also use the <i>backup-file</i> argument to specify a file on a local or remote disk. If <i>backup-file</i> is '-', the files are read from standard input.
<code>h</code>	Turns off directory expansion. Only the directory file you specify is extracted or listed.
<code>m</code>	Restores specified files into the current directory on the disk regardless of where they are located in the backup hierarchy and renames them with their inode number. For example, if the current working directory is <code>/files</code> , a file in the backup named <code>./dready/fcs/test</code> with inode number 42, is restored as <code>/files/42</code> . This option is useful only when you are extracting a few files.
<code>s</code> <i>n</i>	Skips to the <i>n</i> th backup file on the media. This option is useful when you put more than one backup on a single tape.
<code>v</code>	Verbose. Displays the names and inode numbers of each file as it is restored.
<code>y</code>	Continues when errors occur reading the media and tries to skip over bad blocks instead of stopping and asking whether to abort. This option tells the command to assume a yes response.

Interactive Commands

When you use the interactive command, a `restore >` prompt is displayed, as shown in this example:

```
jupiter% ufsrestore ivf /dev/rmt/0
Verify volume and initialize maps
Tape block size is 126
Dump date: Tue Aug 18 09:06:43 1989
Dumped from: Fri Aug 14 08:25:10 1989
Level 0 dump of /usr on pilgrim:/dev/dsk/c0t1d0s6
Label:none
Extract directories from tape
Initialize symbol table.
restore >
```

At the `restore >` prompt, you can use the commands shown in Table 8-3 to find files, create a list of files to be restored, and restore them.

Table 8-3 Commands for Interactive Restore

Option	Description
<code>ls [directory-name]</code>	Lists the contents of either the current directory or the specified directory. Directories are marked by a / suffix and entries in the current list to be restored (extracted) are marked by an * prefix. Inode numbers are shown if the verbose option is used.
<code>cd directory-name</code>	Changes to the specified directory in the backup hierarchy.
<code>add [filename]</code>	Adds the current directory or the specified file or directory to the list of files to extract (restore). If you do not use the <code>h</code> option, all files in a specified directory and its subdirectories are added to the list. Note that all the files you want to restore to a directory might not be on a single backup tape or diskette. You might need to restore from multiple backups at different levels to get all the files.
<code>delete [filename]</code>	Deletes the current directory or the specified file or directory from the list of files to extract (restore). If you do not use the <code>h</code> option, all files in the specified directory and its subdirectories are deleted from the list. Note that the files and directories are deleted only from the extract list you are building. They are not deleted from the media.

Table 8-3 Commands for Interactive Restore (Continued)

Option	Description
<code>extract</code>	Extracts the files in the list and restores them to the current working directory on the disk. Specify 1 when asked for a volume number. If you are doing a multitape or multidiskette restore and restoring a small number of files, start with the last tape or diskette.
<code>help</code>	Displays a list of commands you can use in interactive mode.
<code>pwd</code>	Displays the path name of the current working directory in the backup hierarchy.
<code>q</code>	Quits interactive mode without restoring any additional files.
<code>setmodes</code>	Lets you set the mode for files to be restored to match the mode of the root directory of the file system from which they were backed up. You are prompted with: <code>set owner/mode for '.' [yn]?</code> Type <code>y</code> (for yes) to set the mode (permissions, owner, times) of the current directory to match the root directory of the file system from which they were backed up. Use this mode when restoring a whole file system. Type <code>n</code> (for no) to leave the mode of the current directory unchanged. Use this mode when restoring part of a backup to a directory other than the one from which the files were backed up.
<code>verbose</code>	Turns on or off the verbose option (which can also be entered as <code>v</code> on the command line outside of interactive mode). When verbose is on, the interactive <code>ls</code> command lists inode numbers and the <code>ufsrestore</code> command display information on each file as it is extracted.
<code>what</code>	Displays the backup header on the tape or diskette.


Part 3 — Configuring Swap Space and Managing Disk Use

This part has two chapters:

Chapter 9, “Configuring Additional Swap Space,” explains how to add additional swap space without reconfiguring a disk.

Chapter 10, “Managing Disk Use,” explains how to monitor disk use, manage disk quotas, and monitor and remove large files.

Configuring Additional Swap Space

9 

This chapter has these sections:

<i>Swap Space and Virtual Memory</i>	<i>page 160</i>
<i>Swap Space and the tmpfs File System</i>	<i>page 161</i>
<i>How Do I Know If I Need More Swap Space?</i>	<i>page 162</i>
<i>How Swap Space is Allocated</i>	<i>page 163</i>
<i>Planning for Swap Space</i>	<i>page 164</i>
<i>Monitoring Swap Resources</i>	<i>page 165</i>
<i>Adding More Swap Space</i>	<i>page 166</i>
<i>Removing a Swap File From Use</i>	<i>page 168</i>

About Swap Space

This section provides a conceptual overview of swap space and briefly discusses the differences between Solaris 1.x (SunOS 4.x) and Solaris 2.x (SunOS 5.x) swap requirements. If you are already familiar with the Solaris 2.x swap mechanism, proceed to the section called “Planning for Swap Space” on page 164.

It is important for administrators to understand the features of the Solaris 2.x swap mechanism in determining:

- Swap space requirements
- The relationship with the `tmpfs` file system
- Recovery from error messages related to swap space

Swap Space and Virtual Memory

The Solaris 2.x system software uses some disk slices for temporary storage rather than for file systems. These slices are called *swap* slices. Swap slices are used as virtual memory storage areas when the system does not have enough physical memory to handle current processes.

The Solaris 2.x virtual memory system maps physical copies of files on disk to virtual addresses in memory. Physical memory pages which hold the data for these mappings can be backed by regular files in the file system or by swap space. If the memory is backed by swap space it is referred to as *anonymous* memory because the user doesn't know the names of the files backing the memory.

Solaris 1.x anonymous memory pages are mapped using randomly assigned names from the system's swap space pool. These memory pages are used for:

- Private copies of data created during copy-on-write operations
- Process and stack segments
- The `tmpfs` file system storage resources

The limitations of the Solaris 1.x anonymous memory implementation are:

- Physical storage (disk-backed swap) must always be reserved for anonymous memory mappings even if the application doesn't use it.

For example, applications with large data segments must be configured with lots of swap space even if the pages are not written out to physical storage.

- The formula used to associate an anonymous memory page with physical storage is limited and inflexible because the backing store is chosen at random and can never be changed.

The Solaris 2.x software environment introduces the concept of *virtual swap space*, a layer between anonymous memory pages and the physical storage (or disk-backed swap space) that actually back these pages. A system's virtual swap space is equal to the sum of all its physical (disk-backed) swap space plus a portion of the currently available physical memory.

Virtual swap space has these advantages:

- The need for large amounts of physical swap space is reduced because virtual swap space does not necessarily correspond to physical (disk) storage.
- A pseudo file system called `swapfs` provides addresses for anonymous memory pages. Because `swapfs` controls the allocation of memory pages, it has greater flexibility in deciding what happens to a page. For example, it may change the page's requirements for disk-backed swap storage.

Swap Space and the `tmpfs` File System

The `tmpfs` file system is activated automatically in the Solaris 2.x environment by an entry in the `/etc/vfstab` file. The `tmpfs` file system stores files and their associated information in memory (in the `/tmp` directory) rather than on disk, which speeds access to those files. This results in a major performance enhancement for applications such as compilers and DBMS products that use `/tmp` heavily.

The `tmpfs` file system allocates space in the `/tmp` directory from the system's swap resources. This means that as you use up space in `/tmp`, you are also using up swap space. So if your applications use `/tmp` heavily and you do not monitor swap space usage, your system could run out of swap space.

Use the following if you want to use `tmpfs` but your swap resources are limited:

- Mount the `tmpfs` file system with the `size` option (`-o size`) to control how much of the swap resources `tmpfs` can use.
- If you are close to running out of swap space, you can use your compiler's `TMPDIR` environment variable to point to a larger temporary directory.

Using your compiler's `TMPDIR` variable only controls whether the compiler is using `/tmp` or not. It has not effect on other programs' use of `/tmp`.

How Do I Know If I Need More Swap Space?

This section lists several possible error messages displayed when you run out of swap space.

Swap-Related Error Messages

These messages indicate that an application was trying to get more anonymous memory and there was no swap space left to back it.

```
application is out of memory
```

```
malloc error 0
```

```
WARNING: Sorry, no swap space to grow stack for pid
```

tmpfs-Related Error Messages

```
directory: File system full, swap space limit exceeded
```

This message is displayed if a page could not be allocated when writing a file. This can occur when `tmpfs` tries to write more than it is allowed or if currently executed programs are using a lot of memory.

```
directory: File system full, memory allocation failed
```

This messages means `tmpfs` ran out of physical memory while attempting to create a new file or directory.

See `tmpfs(7)` for information on recovering from the `tmpfs`-related error messages.

How Swap Space is Allocated

Initially, swap space is allocated as part of the Solaris installation process. If you use the installation program's automatic layout of disk slices and do not manually change the size of the swap slice, the Solaris installation program allocates default swap slices as shown in Table 9-1.

Table 9-1 Default Swap Space Allocations

If Your System Has <i>n</i> Mbytes of Physical Memory...	Then the Default Swap Space Allocated is ...
16-64	32 Mbytes
64-128	64 Mbytes
128-512	128 Mbytes
greater than 512	256 Mbytes

The /etc/vfstab File

After the system is installed, swap slices and files are listed in the `/etc/vfstab` file and are activated by the `/sbin/swapadd` script when the system is booted.

An entry for a swap device in the `/etc/vfstab` file contains:

- The full path name of the swap slice or file
- File system type of `swap`

Because the file system containing a swap file must be mounted before the swap file is activated, make sure that the entry that mounts the file system comes before the entry that activates the swap file in the `/etc/vfstab` file.

Planning for Swap Space

The most important factors in determining swap space size are the requirements of the system's software applications.

For example, large applications such as computer-aided-design simulators, database-management products, transaction monitors, and geologic analysis systems can consume as much as 200-1000 Mbytes of swap space in very large memory systems.

Consult your application vendor for swap space requirements for any application whose data files typically exceed 10-20 Mbytes in size.

If you are unable to determine swap space requirements from the application vendor, use the following guidelines to allocate swap space:

- To support your applications, allocate:
 - 1 Mbyte per trivial application such as `xterm`.
 - 2-3 Mbytes per lightweight application such as a calendar or mail application.
 - 20-50 Mbytes for large applications such as desktop publishing software.
- To save crash dumps, allocate 100% of physical memory to save a worst-case crash dump.
- If you are unsure of system or application requirements, allocate 50 to 100% of the system's physical memory. For example, allocate 16-32 Mbytes of swap space for a system with 32 Mbytes of physical memory. This will provide 48-64 Mbytes of total virtual swap space.
- Determine whether large applications (like compilers) will be using the `/tmp` directory. Then allocate additional swap space to be used by `tmpfs`. See "Swap Space and the `tmpfs` File System" for information about `tmpfs`.

Monitoring Swap Resources

The `/usr/sbin/swap` command is used to manage swap areas. Two options, `-l` and `-s`, are used to display information about swap resources.

- ◆ Use the `swap -l` command to identify a system's swap areas. Activated swap devices or files are listed under the `swapfile` column.

```
# swap -l
swapfile          dev  swaplo blocks  free
/dev/dsk/c0t2d0s1 32,17      8 205624 192704
```

- ◆ Use the `swap -s` command to monitor swap resources.

```
# swap -s
total: 10492k bytes allocated + 7840k reserved = 18332k used, \
21568k available
```

The `used` plus `available` figures equals total swap space on the system, which includes a portion of physical memory and swap devices (or files).

You can use the amount of swap space available and used (in the `swap -s` output) as a way to monitor swap space usage over time. If a system's performance is good, use `swap -s` to see how much swap space is available. When the performance of a system slows down, check the amount of swap space available to see if it has decreased. Then you can identify what changes to the system might have caused swap space usage to increase. See *Security, Performance, and Accounting Administration* for more information on troubleshooting memory and swap space problems.

Keep in mind when using this command that the amount of physical memory available for swap usage changes dynamically as the kernel and user processes lock down and release physical memory.

Note – The `swap -l` command displays swap space in 512-byte blocks and the `swap -s` command displays swap space in 1024-byte blocks. If you add up the blocks from `swap -l` and convert them to Kbytes, it will be less than `used + available` (in the `swap -s` output) because `swap -l` does not include physical memory in its calculation of swap space.

The output from the `swap -s` command is summarized in Table 9-2.

Table 9-2 Output of the `swap -s` Command

Keyword	Description
bytes allocated	The total amount of swap space in 1024-byte blocks that is currently allocated as backing store (disk-backed swap space).
reserved	The total amount of swap space in 1024-byte blocks not currently allocated, but claimed by memory for possible future use.
used	The total amount of swap space in 1024-byte blocks that is either allocated or reserved.
available	The total amount of swap space in 1024-byte blocks that is currently available for future reservation and allocation.

Adding More Swap Space

As system configurations change and new software packages are installed, you might need to add more swap space. The preferred way to add more swap space is to use the `mkfile` and `swap` commands to designate a part of an existing UFS or NFS file system as a supplementary swap area. These commands, described below, enable you to add more swap space without repartitioning a disk.

An alternative way to add more swap space is to repartition a disk. See the *Administration Supplement for Solaris Platforms* manual for your system for information on how to repartition a disk.

Creating a Swap File

The following steps are used to create a swap file:

- Create a swap file using the `mkfile` command.
- Activate the swap file with the `swap` command.
- Add an entry for the swap file in the `/etc/vfstab` file so that its activated automatically when the system is booted.

The `mkfile` Command

The `mkfile` command creates a file that is suitable for use either as an NFS-mounted or local swap area. The sticky bit is set, and the file is padded with zeros. You can specify the size of the swap file in bytes (the default) or in kilobytes, blocks, or megabytes using the `k`, `b`, or `m` suffixes, respectively.

Table 9-3 shows the options to the `mkfile` command.

Table 9-3 Options to the `mkfile` Command

Option	Description
<code>-n</code>	Creates an empty file. The size is noted, but the disk blocks are not allocated until data is written to them.
<code>-v</code>	Verbose. Reports the names and sizes of created files.

Caution – Use the `-n` option only when creating an NFS swap file.

▼ How to Create a Swap File and Make It Available

1. Become superuser.

You can create a swap file without root permissions, but it is a good idea to have root be the owner of the swap file so that the swap file cannot be inadvertently overwritten.

2. Type `mkfile nnn[k|b|m] filename` and press Return.

The swap file of the size (`nnn`) and file name you specify is created.

In this example, a 20-Mbyte swap file named `swapfile` is created on a UFS file system named `/files`:

```
# mkfile 20m /files/swapfile
```

3. Activate the swap file. Type `/usr/sbin/swap -a /path/filename` and press Return.

You must use the absolute path name to specify the swap file. The swap file is added and available until the file system is unmounted or the system is rebooted.

4. Type `/usr/sbin/swap -l` to verify that the swap file is added.

```
# swap -a /files/swapfile
# swap -l
swapfile          dev  swaplo  blocks  free
/dev/dsk/c0t2d0s1 32,17    8 205624 192704
/files/swapfile   -         8  40952  40952
```

5. Add an entry for the swap file to the `/etc/vfstab` file that specifies the full path name of the file, and designates `swap` as the file system type. For example, to make the swap file `/files/swapfile` available, add this entry to the `/etc/vfstab` file:

```
/files/swapfile      -      -      swap      -      no      -
```

Removing a Swap File From Use

If the user no longer needs the extra swap space, you can remove it.

▼ How to Remove Extra Swap Space

1. Become superuser.

2. Type `/usr/sbin/swap -d /path/filename` and press Return.

The swap file name is removed from the list so that it is no longer available for swapping. The file itself is not deleted.

```
# swap -d /files/swapfile
# swap -l
swapfile          dev  swaplo  blocks  free
/dev/dsk/c0t2d0s1 32,17    8 205624 192720
# ls -l /files/swapfile
-rw-----T  1 root  staff  20971520  Jun 2 12:54 /files/swapfile
#
```

3. Edit the `/etc/vfstab` file and delete the entry for the swap file.

-
4. **Recover the disk space so that you can use it for something else.**
 - a. **If the swap space is a file, type `rm swap-filename` and press Return.**

or
 - b. **If the swap space is on a separate slice and you are sure you will not need it again, make a new file system and mount the slice.**

See “Creating a UFS File System on a Disk Slice” on page 35 and Chapter 3, “Mounting and Unmounting File Systems,” for more information.

This chapter contains these sections:

<i>Monitoring Available Disk Space</i>	<i>page 171</i>
<i>Monitoring Files and Directories</i>	<i>page 176</i>
<i>Reducing Overloaded File Systems</i>	<i>page 186</i>
<i>Controlling Disk Space With Quotas</i>	<i>page 193</i>

You must routinely monitor UFS file systems on disk slices so that performance remains satisfactory. This chapter describes how to monitor available disk space and the size of files and directories, reduce overloaded file systems, and control disk space with quotas.

Monitoring Available Disk Space

You should routinely monitor disk space to see how close to capacity a system is running. Use the `df` command to monitor available disk space. When file systems are nearing maximum capacity, see “Reducing Overloaded File Systems” on page 186 for information on how to make more disk space available.

The `df` command displays information about specific disk slices, including:

- Total number of 512-byte blocks (or kilobytes) and files allocated to each file system
- Number of 512-byte blocks (or kilobytes) and files used in each file system
- Number of 512-byte blocks (or kilobytes) and files remaining in each file system

Using the `df` Command

If you use the `df` command with no arguments, the number of 512-byte blocks used and the number of files for all mounted file systems is displayed. You can run `df` on a specific disk slice by specifying the special device name for the slice (for example, `df /dev/dsk/c0t0d0s0`), the mount point for the file system (for example, `df /usr`), or the name of a directory in the file system. Use the `-F ufs` option to limit the information to local UFS file systems. To display information about unmounted file systems not in the `/etc/vfstab` file, specify the file system name and type. Table 10-1 shows the most frequently used options to the `df` command.

Table 10-1 Options for the `df` Command

Option	Description
<code>-t</code>	Displays total 512-byte blocks and files allocated and number of 512-byte blocks and files used
<code>-g</code>	Displays total 512-byte blocks and files allocated, used, and free, as well as the type of file system, file system ID, file name length, block size, and fragment size
<code>-k</code>	Displays the used kilobytes, free kilobytes, and the percent of capacity used

Note – For remotely mounted file systems, `-1` is displayed instead of the number of files.

See the `df(1M)` manual page for a complete list of options.

▼ **How to Display Blocks and Files Used for All Mounted File Systems**

◆ **Type `df` and press Return.**

A list of mounted file systems, device names, total 512-byte blocks used, and number of files is displayed.

In this example, `/`, `/usr`, `/proc`, and `/tmp` are on the local disk. The other file systems are NFS-mounted and do not use local disk resources:

```
saturn% df
/                (/dev/dsk/c0t0d0s0):  21338 blocks   9592 files
/usr             (/dev/dsk/c0t0d0s6):  46722 blocks  34103 files
/proc           (/proc              ):    0 blocks     112 files
/tmp            (swap               ): 66696 blocks  3177 files
/root           (saturn:(pid132)):    0 blocks     -1 files
/home           (saturn:(pid132)):    0 blocks     -1 files
/src            (saturn:(pid132)):    0 blocks     -1 files
/nse            (saturn:(pid132)):    0 blocks     -1 files
/net            (saturn:(pid132)):    0 blocks     -1 files
saturn%
```

▼ How to Display Used Disk Space in Kilobytes and Its Percent of Capacity

◆ Type `df -k` and press Return.

The file system, total kilobytes, used kilobytes, available kilobytes, percent of capacity used, and mount point are displayed:

```
saturn% df -k
filesystem      kbytes  used   avail   capacity  mounted on
/dev/dsk/c0t0d0s0 22199  11530  8459    58%      /
/dev/dsk/c0t0d0s6 73399  50038  16031   76%      /usr
/proc           0       0       0       0%       /proc
swap           33364   8       33356   0%       /tmp
saturn:(pid132) 0       0       0       0%       /root
saturn:(pid132) 0       0       0       0%       /home
saturn:(pid132) 0       0       0       0%       /src
saturn:(pid132) 0       0       0       0%       /nse
saturn:(pid132) 0       0       0       0%       /net
saturn%
```

▼ How to Display Blocks and Files Used for UFS File Systems

◆ Type `df -F ufs` and press Return.

A list of UFS file systems, device names, total 512-byte blocks used, and number of files is displayed.

This example is for the same system as the previous example. This command shows only the UFS file systems. Although `/proc` and `/tmp` local file systems, they are not UFS file systems (`/proc` is a PROCFS file system, and `/tmp` is a TMPFS file system):

```
saturn% df -F ufs
/                (/dev/dsk/c0t0d0s0): 21338 blocks 9592 files
/usr             (/dev/dsk/c0t0d0s6): 46722 blocks 34103 files
saturn%
```


▼ **How to Display Blocks and Files Used and Total Blocks and Files for All Mounted File Systems**

◆ **Type `df -t` and press Return.**

A list of all mounted file systems, device names, total 512-byte blocks used, and number of files is displayed. The second line displays the total number of blocks and files allocated for the file system:

```
saturn% df -t
/                (/dev/dsk/c0t0d0s0):    21338 blocks   9592 files
                total:        44398 blocks   11264 files
/usr            (/dev/dsk/c0t0d0s6):    46722 blocks   34103
files
                total:        146798 blocks  37888 files
/proc          (/proc                ):    0 blocks      112 files
                total:        0 blocks      140 files
/tmp           (swap                ):    66712 blocks  3177 files
                total:        66728 blocks  3179 files
/root          (saturn:(pid132)):    0 blocks      -1 files
                total:        0 blocks      -1 files
/home         (saturn:(pid132)):    0 blocks      -1 files
                total:        0 blocks      -1 files
/src          (saturn:(pid132)):    0 blocks      -1 files
                total:        0 blocks      -1 files
/nse         (saturn:(pid132)):    0 blocks      -1 files
                total:        0 blocks      -1 files
/net          (saturn:(pid132)):    0 blocks      -1 files
                total:        0 blocks      -1 files
saturn%
```

Monitoring Files and Directories

When you have determined (using the `df` command) which file systems are overloaded, you can use the commands described in Table 10-2 to monitor files and directories.

Table 10-2 Commands Used to Monitor Files and Directories

Command	Information Provided
<code>ls</code>	Size, age, permissions, owner of files
<code>du</code>	Total size of directories and their contents
<code>quot</code>	Number of blocks owned by users
<code>find</code>	Names of files meeting search criteria

These sections tell you how to look for potential problems:

- “Monitoring System Log Files That Grow” on page 176
- “Finding Large Files” on page 178
- “Finding Large Directories” on page 180
- “Finding Large Space Users” on page 182
- “Finding Old and Inactive Files” on page 185

Monitoring System Log Files That Grow

As part of normal system operation, some system log files can grow quite large. Most of these log files are located in the `/var` directory. Table 10-3 lists examples of some of the system administrative files that can grow daily.

Table 10-3 System Log Files That Grow

File	Use
<code>/var/adm/aculog</code>	Log of outgoing modem calls
<code>/var/adm/admin.log</code>	Log of Administration Tool activities
<code>/var/adm/lastlog</code>	History of last logins
<code>/var/adm/messages</code>	Messages from <code>syslogd</code>
<code>/var/adm/pacct</code>	Per process accounting information (if accounting is turned on)
<code>/var/adm/sa/*</code>	System accounting files
<code>/var/adm/sulog</code>	History of <code>su</code> commands

Table 10-3 System Log Files That Grow (Continued)

File	Use
/var/adm/utmp	History of user logins
/var/adm/wtmp	History of system logins
/var/cron/log	History of actions of /usr/sbin/cron
/var/spell/spellhist	Words that spell(1) fails to match
/var/lp/logs/*	LP print service logs

Note – If your system is running accounting or system activity reporting (*sar*), see *Security, Performance, and Accounting Administration* for a list of associated files that you should also monitor.

▼ How to Check the Size of Files (in Kilobytes)

1. Type `cd directory` and press Return.

2. Type `ls -l` and press Return.

A list of the files and directories is displayed, showing the size in kilobytes.

In this example, you can see that `lastlog`, `wtmp`, and `wtmpx` are substantially larger than the other files in the `/var/adm` directory.

Code Example 10-1 Checking the Size of Files

```

venus% cd /var/adm
venus% ls -l
total 434
-r--r--r--  1 root    other    585872 Jan 28 14:53 lastlog
drwxrwxr-x  2 adm     adm      512 Dec  1 16:35 log
-rw-r--r--  1 root    other    408 Jan 28 14:15 messages
-rw-r--r--  1 root    other    177 Jan 24 16:56 messages.0
-rw-r--r--  1 root    other    177 Jan 17 16:13 messages.1
-rw-r--r--  1 root    other     0 Jan  4 04:05 messages.2
-rw-r--r--  1 root    other    562 Jan  2 13:13 messages.3
drwxrwxr-x  2 adm     adm      512 Dec  1 16:35 passwd
drwxrwxr-x  2 adm     sys      512 Jan 28 11:38 sa
-rw-rw-rw-  1 bin     bin       0 Nov 26 10:56 spellhist
-rw-----  1 root    root    1319 Jan 28 14:58 sulog
-rw-r--r--  1 root    bin      288 Jan 28 14:53 utmp

```

Code Example 10-1 Checking the Size of Files (Continued)

```
-rw-r--r--  1 root    bin           2976 Jan 28 14:53 utmpx
-rw-rw-r--  1 adm     adm           12168 Jan 28 14:53 wttmp
-rw-rw-r--  1 adm     adm          125736 Jan 28 14:53 wttmpx
venus%
```

▼ How to Check the Size of Files (in Blocks)

1. Type `cd` *directory* and press Return.

2. Type `ls -s` and press Return.

A list of the files and directories is displayed, showing the size in blocks.

In this example, you can see that lpNet uses eight blocks and lpsched and lpsched-1 use two blocks each.

```
venus% cd /var/lp/logs
venus% ls -s
total 14          2 lpsched-1      0 lpsched-4      0 requests-2
   8 lpNet        2 lpsched-2      0 requests
   2 lpsched      0 lpsched-3      0 requests-1
venus%
```

See “Reducing Overloaded File Systems” on page 186 for information on how to reduce the size of system log files.

Finding Large Files

When you need more space archiving or moving large files obviously recovers more space than moving small files. This section provides some suggestions for commands you can use to locate large files.

▼ How to Find Large Files (in Blocks)

1. Type `cd directory` and press Return.

2. Type `ls -s | sort -nr | more` and press Return.
The files are sorted by block size from largest to smallest:

```
venus% cd /var/adm
venus% ls -s | sort -nr | more
total 624
 320 wtmpx
 128 lastlog
  74 pacct
  56 messages
  30 wtmp
   6 utmpx
   2 utmp
   2 sulog
   2 sa
   2 passwd
   2 log
   0 spellhist
venus%
```

▼ How to Find Files That Exceed a Given Size Limit

♦ Type `find directory -size +nnn -print` and press Return.

Files above the size you specify (in 512-byte blocks) are listed.

This example shows how to find files with more than 400 (512-byte) blocks in the current working directory:

Code Example 10-2 Finding Files That Exceed a Given Limit

```
venus% find . -size +400 -print
/home/jones/Howto/howto.doc
/home/jones/Howto/howto.doc.backup
/home/jones/Howto/howtotest.doc
/home/jones/Routine/routineBackupconcepts.doc
/home/jones/Routine/routineIntro.doc
/home/jones/Routine/routineTroublefsck.doc
```

Code Example 10-2 Finding Files That Exceed a Given Limit (Continued)

```
/home/jones/.record  
/home/jones/Mail/pagination  
/home/jones/Config/configPrintadmin.doc  
/home/jones/Config/configPrintsetup.doc  
/home/jones/Config/configMailappx.doc  
/home/jones/Config/configMailconcepts.doc  
/home/jones/snapshot.rs  
venus%
```

Finding Large Directories

Use the `du` command to find large directories. It provides summary totals on directories without listing every file. You can use the `du` command to get the total size (in blocks) of one or more directories (summing all the files and directories below), the total sizes of each subdirectory, and even the sizes of each file. The `du` command does not follow symbolic links, and it works only on locally mounted UFS file systems.

If you use the `du` command with no options, it displays the total block count for the specified directories and the subtotals for each subdirectory. Table 10-4 shows the options for the `du` command.

Table 10-4 Options for the du Command

Option	Description
<code>-s</code>	Displays a total block count for the specified directories only, which sums up everything beneath
<code>-a</code>	Displays the total block count for the specified directories, the subtotals for each subdirectory, and the size of each file

Note – You can run `du` on individual files using the `-a` or `-s` option. No size is reported if you run the command on a file without using an option.

▼ **How to Display the Total Size (in 512-byte Blocks) of One or More Directories**

♦ **Type `du -s directory directory ...` and press Return.**

The sizes of the directory or directories are displayed in 512-byte blocks:

```
venus% du -s /usr/sbin /var/adm
12672 /usr/sbin
574 /var/adm
venus%
```

▼ **How to Display the Total Size (in 512-byte Blocks) of Directories and Their Subdirectories**

♦ **Type `du directory directory ...` and press Return.**

The sizes of the directory or directories and all subdirectories are displayed in 512-byte blocks:

```
venus% du /usr/sbin /kernel
2546 /usr/sbin/static
12672 /usr/sbin
2108 /kernel/drv
36 /kernel/exec
1714 /kernel/fs
458 /kernel/misc
62 /kernel/sched
418 /kernel/strmod
94 /kernel/sys
4892 /kernel
venus%
```

▼ **How to Display the Total Size (in 512-byte Blocks) of One or More Directories, All Subdirectories, and All Files**

♦ **Type `du -a directory directory ...` and press Return.**

The sizes of the directory or directories, subdirectories, and all files are displayed in 512-byte blocks:

```
venus% du -a /usr/sbin /kernel
18    /usr/sbin/add_drv
16    /usr/sbin/arp
240   /usr/sbin/autopush
8     /usr/sbin/chroot
22    /usr/sbin/ckbupscd
18    /usr/sbin/clri
58    /usr/sbin/cron
2     /usr/sbin/dcopy
42    /usr/sbin/devlinks
2     /usr/sbin/devnm
30    /usr/sbin/df
10    /usr/sbin/dfshares
32    /usr/sbin/disks
12    /usr/sbin/dispadm
18    /usr/sbin/dname
20    /usr/sbin/drvconfig
2     /usr/sbin/drvload
2     /usr/sbin/EEPROM
(More files not shown in this example)
venus%
```

Finding Large Space Users

At times, you will need to find out who is using the most disk space. Type `du -s user's home directory` to display a summary of the space occupied by each user.

You can also use the `quot` command to find out who owns the space in any local UFS file system.

Note – The `quot` command works only on local UFS file systems. If you use the `quot` command on another file system, the error message `filesystem not ufs filesystem` is displayed.

▼ **How to Display the User Allocation of a UFS File System**

1. Become superuser.

2. Type `quot filesystem` and press Return.

The users of a file system and the number of 1024-byte blocks used are displayed.

In this example, the users of the root file system are displayed:

```
mars% su
Password:
# quot /
/dev/dsk/c0t0d0s0 (st):
8130      root
2451      bin
143       adm
109       lp
47        uucp
19        sys
7         pubs
1         svvs
#
```

▼ How to Display Users for All UFS-Mounted File Systems**1. Become superuser.****2. Type `quot -a` and press Return.**

All users of a file system and the number of 1024-byte blocks used are displayed:

```
mars% su
Password:
# quot -a
/dev/rdisk/c0t0d0s0 (/):
 8130      root
 2451      bin
  143      adm
  109      lp
   47      uucp
   19      sys
    7      pubs
    1      svvs
/dev/rdisk/c0t0d0s6 (/usr):
41004      bin
15274      root
 2248      lp
  759      uucp
    1      adm
    1      sys
/dev/rdisk/c0t0d0s7 (/opt):
78724      root
37409      bin
/dev/rdisk/c0t3d0s7 (/export/home):
  23      pubs
   9      root
#
```

Finding Old and Inactive Files

Part of the job of cleaning up heavily loaded file systems involves locating and removing files that have not been used recently.

Use the `ls -t` command on a directory to list the most recently created or changed files first.

Use the `find(1)` command to locate files that have not been accessed for a specified amount of time. For example, use this command to locate regular files in the `/home` directory that have not been accessed in the last 60 days and to record the list in a file:

```
# find /home -type f -atime +60 -print > /var/tmp/deadfiles &
#
```

Here is an example of the output from the `find` command:

```
venus% more /var/tmp/deadfiles
/home/jones/cat.rs
/home/jones/cmdayprint.ps
/home/jones/agenericwindow.rs
/home/jones/bindericon.rs
/home/jones/binderwindow.rs
/home/jones/blankkeys.tmp
/home/jones/calcalpha keypad.rs
/home/jones/calcangle setting.rs
/home/jones/calcbinaryfuns.rs
/home/jones/calcicon.rs
/home/jones/calclogfuns.rs
/home/jones/calcllogicalfuns.rs
/home/jones/calcmemoryreg.rs
/home/jones/calcmiscfuns.rs
(More files not shown in this example)
venus%
```

Reducing Overloaded File Systems

When you have determined which file systems are nearing capacity, you then need to determine the best way to free up additional disk space. This section describes how to free additional space in these ways:

- “Truncating Files That Grow” on page 186
- “Deleting Old or Inactive Files” on page 187
- “Clearing Out Temporary and Obsolete Files” on page 188
- “Deleting core Files” on page 189
- “Removing Crash Dump Files” on page 190
- “Creating Links Instead of Duplicating Files” on page 191
- “Moving Directory Trees Between File Systems” on page 192

Truncating Files That Grow

A good way to limit the size of files that grow is to periodically truncate them, keeping only the most recent lines at the end of the file.

▼ How to Truncate a File

1. **Type `ls -l filename` and press Return.**
2. **Record the permissions, user and group ownership, and size of the file you want to truncate.**
3. **Become superuser.**
4. **Type `tail log-name > filename` and press Return.**
The last 10 lines (by default) of the log file are saved in a temporary file. If you want to save more than the last 10 lines, type the number of lines as an option. For example, to save 50 lines, type `tail -50 log-name > filename`.
5. **Type `mv filename log-name` and press Return.**
The old log is replaced by the shorter one.
6. **Type `ls -l log-name` and press Return.**
The permissions, user and group ownership, and size of the log file are displayed.

7. Check to be sure the permissions and group ownership have not been changed. If they have, change them back to the way they were.

If you move the files as `root`, the group ownership may have been changed and members of the group will not be able to access the files until you change the group ownership back to its original value.

In this example, `/var/adm/sulog` is truncated to contain the 50 most recent entries:

```
venus% su
Password:
# ls -l /var/adm/sulog
-rw----- 1 root  root    585872 Jan 29 16:49 /var/adm/sulog
# tail -50 /var/adm/sulog > /var/tmp/sulog
# mv /var/tmp/sulog /var/adm/sulog
# ls -l /var/adm/sulog
-rw-r--r-- 1 root  other    1424 Jan 29 16:50 /var/adm/sulog
# chmod 600 /var/adm/sulog
# chgrp root /var/adm/sulog
# ls -l /var/adm/sulog
-rw----- 1 root  root    1424 Jan 29 16:50 /var/adm/sulog
#
```

Deleting Old or Inactive Files

You can delete old or inactive files one by one by typing `rm filename` and pressing Return. Alternatively, you can create a file that contains a list of inactive files. Review the list to make sure the files are no longer needed or are backed up so they can be restored if needed later.

▼ How to Delete a List of Inactive Files

1. Become superuser on the server for the file system.

See *NFS Administration Guide* if you need information on NFS security.

2. Type `find directory -type f -mtime +nn -print > /var/tmp/deadfiles &` **and press Return.**

Files older than the time specified (in days) are listed in the file `/var/tmp/deadfiles`.

3. Review the list to be sure all the files should be deleted. Remove the names of the files you want to retain.

4. Type `rm `cat /var/tmp/deadfiles`` and press Return.
The files listed in `/var/tmp/deadfiles` are removed.

In this example, all files not modified for more than 60 days listed in the file `/var/tmp/deadfiles` are removed:

```
pluto% su
Password:
# find /home/jones -type f -mtime +60 -print > /var/tmp/deadfiles &
# rm `cat /var/tmp/deadfiles`
pluto%
```

Clearing Out Temporary and Obsolete Files

The `/tmp` and `/var/tmp` directories store temporary files. By default, the `/tmp` directory is a TMPFS file system, so files in the `/tmp` directory do not take up space in the `/` file system. The files in the `/tmp` directory are deleted each time the file system is unmounted or the system is rebooted. See “The Temporary File System (TMPFS)” on page 4 for more information.

Files in `/var/tmp` take up file system disk space and are retained if the file system is unmounted or the system is rebooted. You should periodically clear out files in `/var/tmp`. In addition, look for obsolete files in the subdirectories of `/var/spool`, such as `/var/spool/mail` and `/var/spool/uucppublic`. If there are some files that need to be saved, delete the files individually by name rather than using the wild card `*`, as shown below.

Note – Unless your site has a specific policy about deleting old `/var/spool/mail` and `/var/spool/uucppublic` files, it is a good idea to check with the owner of these files before deleting them. If you need space in the file system, consider moving the entire file to the user’s home directory and starting a new spooling file.

▼ **How to Clear Out the `/var/tmp` Directory**

1. **Become superuser.**
2. **Type `cd /var/tmp` and press Return.**



Caution – Be sure you are in the right directory before using a potentially destructive command like `rm -r *`.

3. **Type `rm -r *` and press Return.**
All files and subdirectories are deleted.
4. **Change to subdirectories in `/var/spool` (for example, `mail`), look for obsolete files, and delete them.**

Deleting `core` Files

From time to time a program may crash and leave a dump of what was in memory in a file named `core`. These `core` files can be useful to help the company that developed the software product to determine the cause of persistent problems. `core` files may be quite large. When a program crashes repeatedly, it overwrites any existing `core` file with the new information, so that, within any give directory, `core` files do not accumulate. `core` files can, however, be created in different directories depending on which program was running when the dump was done.

Note – Instruct users not to use `core` as a name for any of their files. If users create files named `core`, these files will be overwritten if a program crashes in that directory.

▼ How to Find and Delete `core` Files

- 1. Become superuser.**
- 2. Change to the directory where you want to start the search.**
- 3. Type `find . -name core -exec rm {} \;` and press Return.**
Any `core` files found in the current directory and its subdirectories are removed:

```
pluto% su
Password:
# cd /home/jones
# find . -name core -exec rm {} \;
#
```

Removing Crash Dump Files

Crash dump files are `core` files that contain an image of the state of the operating system kernel at the time that it failed. This image can be used to look at the control structures, active tables, and other relevant information about the operation of the kernel. This information may be useful in reporting or tracking down kernel bugs. Some systems may be set up to automatically save information about kernel crashes in the `/var/crash/system-name` directory. To uniquely identify each crash dump, the files are named sequentially `vmcore.n`. The number *n* is incremented for each subsequent crash dump. If the system is experiencing kernel problems, the size of the crash dump directory can become quite large.

▼ How to Delete Crash Dump Files

- 1. Become superuser.**
- 2. Type `cd /var/crash/system-name` and press Return.**
- 3. Type `rm *` and press Return.**

See *Common Administration Tasks* for more information about crash dumps.

Creating Links Instead of Duplicating Files

Sometimes you need to have the same file in more than one place. For example, when files are moved you may want to retain the old path name so that software that uses an absolute path name can find them. Putting the files in both directories unnecessarily uses disk space. You can use *links* to provide alternative path names for individual files so the data is stored in only one place.

You can create two kinds of links:

- *Symbolic links* can be located anywhere and provide a pointer to the name of the file they link to. Symbolic links can span physical devices and are thus very useful in a networked environment.
- *Hard links* use the same inode number and must be used within the same file system. You cannot, for example, create a hard link between a file in the root directory and a file in a user's home directory.

See the `ln(1)` manual page for more information.

▼ How to Create a Symbolic Link

- ◆ **Type `ln -s source-filename linked-filename` and press Return.**
source-filename is the name of an existing file and *linked-filename* is the name of the file to create as a symbolic link to *source-filename*.

▼ How to Create a Hard Link

- ◆ **Type `ln source-filename linked-filename` and press Return.**
The file named by *source-filename* must exist and must be in the same disk slice (file system) as the file named by *linked-filename*.

Moving Directory Trees Between File Systems

As user needs change and file systems approach maximum capacity, you may need to move user accounts from one file system to another.

▼ How to Move Directory Trees From One File System to Another

- 1. Notify affected users, preferably by email, that you plan to move their files and that path name dependencies may need to be changed.**
- 2. Become superuser.**
- 3. Type `cd /filesystem1` and press Return.**
- 4. Type `find username -print -depth | cpio -pdm /filesystem2` and press Return.**
The files are copied and symbolic links are preserved.
- 5. To verify that the copy was successful, type `cd /filesystem2/username;ls` and press Return.**
- 6. To remove the old directory tree, type `rm -rf /filesystem1/username` and press Return.**
- 7. Use `admintool` to change the `username` default login directory in the **Passwd Database.**
See *User Accounts, Printers, and Mail Administration* for information on how to use the Administration Tool.**

Controlling Disk Space With Quotas

Disk space is always a limited resource. One way you can control available disk space is through the way you set up and allocate available file system space to your users. Another way to control disk space for UFS file systems is to set up and administer *disk quotas*. Disk quotas let you set limits for each user of a file system to control the maximum number of files each user can create and the maximum amount of disk space available to each user.

Each quota has two limits: a *hard limit* and a *soft limit*. Users other than `root` can never exceed the hard limit. When the hard limit is reached, the operating system notifies the user and refuses to allocate any more resources. Users can exceed the soft limit temporarily for a limited period of time. The user receives a warning message, and the operating system allocates the additional resources. If the disk use exceeds the soft limit at the next login, the warning message is repeated. Depending on how quotas are implemented, users who continue to exceed the soft limit will be denied access to additional resources until enough files are deleted to bring disk usage below the soft limit.

If you implement a quota system, you must decide which file systems need quotas. Usually, the file systems that contain user home directories are good candidates for quotas. You do not need to set quotas for `/tmp` file system or for the file systems reserved for public files (for example, the `/` and `/usr` file systems) unless they are writable and can grow.

Quotas have a slight impact on performance, require some administration, and may inconvenience users. Consider implementing quotas only if disk space is very limited and tight security is required at your site.

The quotas utility lets you:

- Set up quotas for each user in a file system
- Turn the enforcement of quotas on and off for each file system
- Report on the quotas defined and the current disk use for each user of each file system
- Synchronize the recorded use with the actual disk use

Setting Up and Administering Quotas

Before you begin to set quotas, you need to do some system configuration. The following steps briefly describe how to set up quotas. See “How to Configure File Systems for Disk Quotas” on page 198 for detailed instructions.

1. Edit `/etc/vfstab` to flag file systems with an `rq` mount option.
2. Create a `quotas` file in the top directory of each file system.
3. Use the quotas editor `edquota` to set up the limits for the quotas.
4. Use `quotaon` to turn on the quotas so they begin to be enforced.

When quotas are set up, you can administer them in these ways:

- Turn quotas on and off for specified file systems
- Change the time limit (that applies to all users)
- Change the quotas (hard and soft limits on blocks and files) for individual users
- Disable quotas for an individual user
- Ensure the consistency of recorded disk use on specified file systems with actual disk use
- Report on the current quotas and disk use for individuals or all users

Table 10-5 describes the commands you use to set up and administer disk quotas.

Table 10-5 Disk Quota Commands

Command	Task
<code>edquota (1M)</code>	Set or change the hard and soft limits on the number of files and/or disk space for each user. In addition, define the time period for each file system that its soft limits can be exceeded by any user.
<code>quot (1M)</code>	Display information for a specified file system on the number of files and 1024-byte blocks owned by each user on the system.
<code>quota (1M)</code>	Display user quotas and current disk use. You can also display user over-quota use.
<code>quotaon (1M)</code>	Turn on (enforce) the quotas for the specified file systems.

Table 10-5 Disk Quota Commands (Continued)

Command	Task
<code>quotaoff(1M)</code>	Turn off quotas for the specified file systems.
<code>quotacheck(1M)</code>	Check the number of blocks and files owned by each user for a given file system and update the information in the quota file so that it is consistent with the current state of the disk.
<code>repquota(1M)</code>	Display the quotas in force for each user and the current number of files and amount of space they own for the specified file systems.

How Quotas Affect Users

When users exceed the soft limit for blocks or inodes, the timer is started. No warning messages are displayed. If users then reduce usage to a level under the soft limit, the timer is turned off. If the user has not reduced usage to an appropriate level when the timer expires, error messages are displayed saying that the file system is full. Any further attempts by the user to acquire more file system resources fail. The messages persist until the user has reduced usage to a level below the soft limit.

Any time users try to exceed the hard limit, an error message is displayed saying that the file system is full. No further disk resources are allocated.

Users can run the `quota` command to find out how much file system space is available and whether they have exceeded the soft limit. Users should be encouraged to include `quota` in their `.profile` or `.cshrc` file so that it is run automatically when they log in.

How Disk Quotas Work

The quotas system uses a file named `quotas` to control available disk space for each file system. The `edquota` command generates information to set up or change the limits for specified users on specified file systems. The information from the `edquota` command is stored in the `quotas` file. The `quotas` file is used by the UFS utilities and the `quotacheck` command to record the actual use of disk space. Each user on the system has an entry in each `quotas` file, whether or not quotas have been imposed on them for that file system. These entries are indexed by user ID and are kept in binary form.

Each time you add or modify quotas for a specified user, `edquota` looks in `/etc/mnttab` to see which UFS file systems are mounted, and it checks the root directory of each file system for a `quotas` file. `edquotas` then creates a temporary text file from the binary `quotas` files. You edit the file and save the changes. The resulting file is then converted back to the binary format `quotas` file.

After quotas are set up, you must turn them on for each file system with the `quotaon` and `quotaoff` commands. The `quotaon` and `quotaoff` commands turn quotas on and off by changing the entry in the `mntopts` field of the mount table `/etc/mnttab`. When the entry is `rq`, it means read/write with quotas in effect. `quotaon` can turn on file systems specified on the command line or turn on all file systems that have `rq` in the `mntopts` field of the `/etc/vfstab` file. The latter option is normally used in a startup script (for example, `/sbin/rc2`) to turn on all quotas when the system boots up.

Figure 10-1 shows how disk quotas operate.

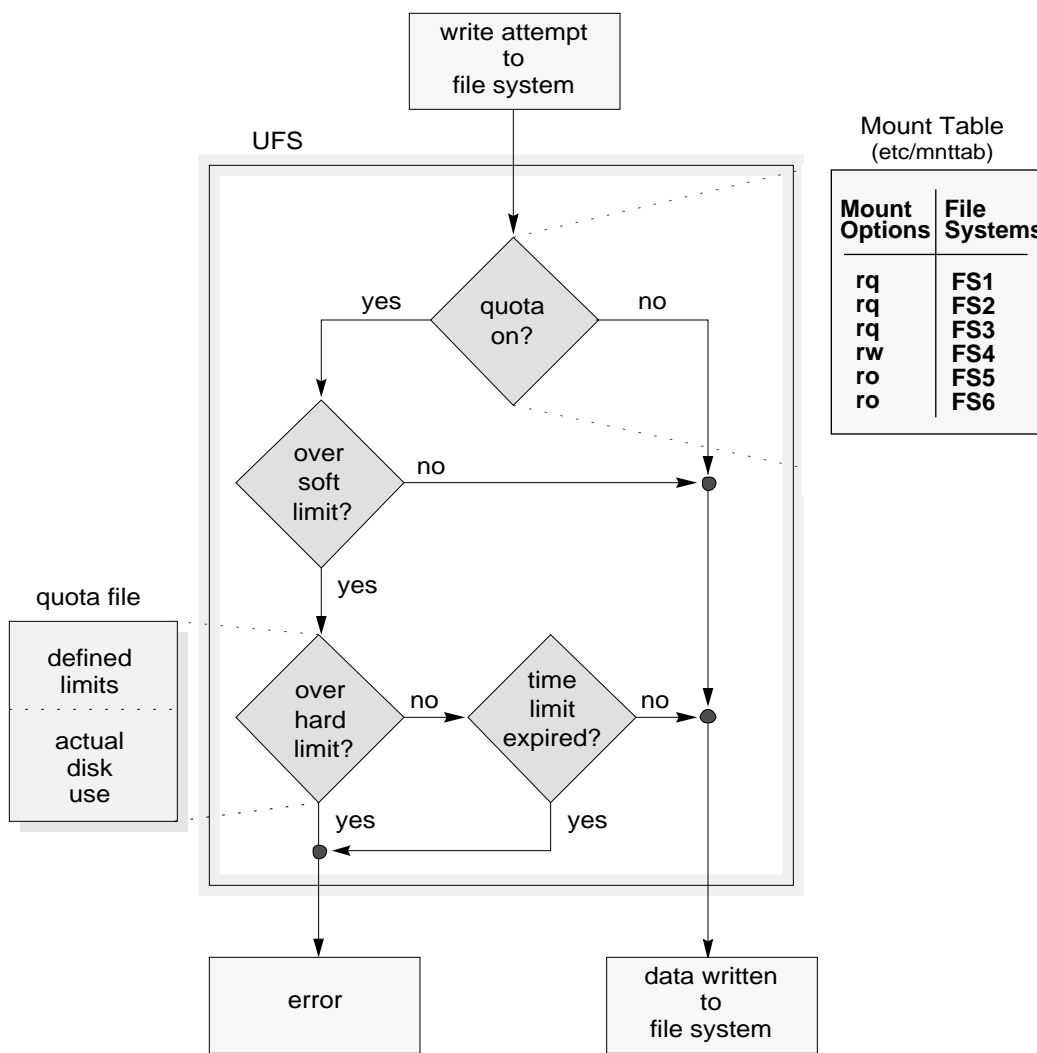


Figure 10-1 Disk Quotas in Operation

▼ How to Configure File Systems for Disk Quotas

You add entries to the `/etc/vfstab` file to activate quotas for specified file systems each time they are mounted, and create a file named `quotas` in the top-level directory of each file system for which you want to use quotas.

1. Become superuser.

2. Edit `/etc/vfstab`, enter `rq` in the `mount options` field for each UFS file system to have quotas, and save the changes.

File systems with `rq` in the `mount options` field have quotas automatically turned on when the system boots.

3. Change to the top directory of the file system.

For example, to turn on quotas for the `/home` file system, type `cd /home` and press Return.

4. Type `touch quotas` and press Return.

A file named `quotas` is created.

5. Type `chmod 600 quotas` and press Return.

Permissions are changed to read/write for root only.

▼ How to Change the Default Time Limit

The default soft limit time is one week. After one week of repeated violations of the soft limit, users cannot access additional system resources. If you want to use a different time limit for a file system, you can change it by following the steps in this section. You can set the time limit only for an entire file system.

1. Become superuser.

2. Type `edquota -t` and press Return.

The default editor opens a temporary file that includes a line for each mounted file system with a `quotas` file in its top directory. For example, if the file system `/files` is the only such mounted file system, this information is displayed:

```
fs /files blocks time limit = 0 (default), files time limit = 0 (default)
```

The `0 (default)` value means that the default time limit of one week is used.

3. Replace the `0` with a number and a unit.

You can specify the units by `month`, `week`, `day`, `hour`, `min`, or `sec`. For example, to change the limit to one day, you could specify either `1 day` or `24 hour`.

4. Exit the file, saving your changes.**▼ How to Set Up User Quotas**

Before you set up user quotas, you need to determine how much space and how many files to allocate to each user. If you want to be sure the total file system space is never exceeded, you can divide the total size of the file system between the number of users. For example, if three users share a 100-Mbyte slice and have equal disk space needs, you could allocate 33 Mbytes to each. In environments where not all users are likely to push their limits, you may want to set individual quotas so that they add up to more than the total size of the file system. For example, if three users share a 100-Mbyte slice, you could allocate 40 Mbytes to each.

If all values are set to `0`, quotas are disabled. Set both soft and hard block values in 1-Kbyte disk blocks, and set both soft and hard values for the total number of files (inodes).

1. Type `edquota username` and press Return.

The default editor opens a temporary file with one line for each mounted file system that has a `quotas` file in its top-level directory. For example, if the file system `/files` is the only one that has a `quotas` file in its top-level directory, this information is displayed:

```
fs /files blocks (soft = 0, hard = 0) inodes (soft = 0, hard = 0)
```

Note – Although you can specify multiple users as arguments to the `edquota` command, the information displayed does not show which user it belongs with, which could create some confusion.

2. Enter the number of 1-Kbyte disk blocks, both soft and hard, and the number of inodes, both soft and hard.**3. Exit the editor, saving your changes.****▼ How to Use a Prototype Quota for Multiple Users**

When you have established quotas for one user, you can use those quotas as a prototype to set the same quotas for other users on the same file system.

1. Become superuser.**2. Type `edquota -p user-with-quotas username1 username2 username3 ...` and press Return.**

The quotas you already established for the prototype user are set for the additional users you specify.

For example, to apply the quotas established for user `ignatz` to users `mercury` and `pluto`, type:

```
edquota -p ignatz mercury pluto
```

▼ How to Initialize Quotas

Once you have configured file systems for quotas and established quotas for each user, you need to run the `quotacheck` command to initialize the quota files and check consistency with the file system before you turn quotas on.

1. Become superuser.

2. Type `quotacheck -a` and press Return.

All file systems with an `rq` entry in the `/etc/vfstab` file are checked and assigned correct initial values.

You can run `quotacheck` on individual file systems by typing `quotacheck /dev/dsk/filesystem` and pressing Return. For example, to check quotas for `/files` on slice `/dev/dsk/c0t4d0s2`, type:

```
quotacheck /dev/dsk/c0t4d0s2
```

▼ How to Turn Quotas On

The quotas you set up with `edquota` are not enforced until you turn them on. You can turn them on and off any time from a command line. You usually turn them on once after you set them up. If you have properly configured the quota files, quotas are automatically turned on each time a system is rebooted and the file systems are mounted.

1. Become superuser.

2. Type `quotaon -a` and press Return.

All file systems with an `rq` entry in the `/etc/vfstab` file are turned on.

You can turn quotas on for individual file systems by typing `quotaon filesystem filesystem ...` and pressing Return. For example, to turn on quotas for `/files` on slice `/dev/dsk/c0t4d0s2` and `/snag` on slice `/dev/dsk/c0t3d0s2`, type:

```
quotaon /dev/dsk/c0t4d0s2 /dev/dsk/c0t3d0s2
```

Administering Disk Quotas

This section describes how to perform these tasks for administering disk quotas:

- Turn quotas on and off for specified file systems
- Change the soft limit time for a file system
- Change the quotas (hard and soft limits on blocks and files) for individual users
- Disable quotas for an individual user
- Ensure the consistency of recorded disk use with actual disk use
- Report on the current quotas and disk use for individual or all users

▼ How to Turn Quotas Off

1. **Become superuser.**

2. **Type** `quotaoff /dev/dsk/cntndnsn` **and press Return.**
Quotas are turned off for the file system you specify.

Alternatively, type `quotaoff -a` and press Return to turn off quotas for all file systems.

▼ How to Change the Soft Limit Time

1. **Become superuser.**

2. **Type** `edquota -t` **and press Return.**

The default editor opens a temporary file that includes a line for each mounted file system with a `quotas` file in its top directory. For example, if the file system `/files` is the only mounted file system, this information is displayed:

```
fs /files blocks time limit = 0 (default), files time limit = 0 (default)
```

The `0 (default)` value means that the default time limit of one week is used.

3. Replace the 0 with a number and a unit.

You can specify the units by month, week, day, hour, min, or sec. For example, to change the limit to one day, you could specify either 1 day or 24 hour.

4. Exit the file, saving your changes.**▼ How to Change Quotas for Individual Users****1. Type `edquota username` and press Return.**

The default editor opens a temporary file with one line for each mounted file system that has a `quotas` file in its top-level directory. For example, if the file system `/files` is the only one that has a `quotas` file in its top-level directory, this information is displayed:

```
fs /files blocks (soft = 0, hard = 0) inodes (soft = 0, hard = 0)
```

Note – Although you can specify multiple users as arguments to the `edquota` command, the information displayed does not show which user it belongs with, which could create some confusion.

2. Enter the number of 1-Kbyte disk blocks, both soft and hard, and the number of inodes, both soft and hard.**3. Exit the file, saving your changes.****▼ How to Disable Quotas for Individual Users**

To disable quotas for individual users, follow the procedure for changing quotas, and change all values to zero.

Note – Be sure you change the values to zero. Do *not* delete the line from the text file.

▼ How to Check Quota Consistency

Information about disk quotas stored in the quota administration files can get out of synchronization with actual disk use. Just as general file system information can become inaccurate over time, abrupt system halts and crashes can degrade the quota information.

When setting up quotas, you should run the `quotacheck` command manually before turning quotas on. It is a good idea to periodically run the `quotacheck` command if systems are rebooted infrequently.

Note – To ensure accurate disk data, file systems should be quiescent when you run the `checkquota` command. You could create a `cron` script to perform this activity. See the `cron(1M)` manual page for more information.

1. Become superuser.

2. Type `quotacheck -a` and press Return.

All file systems with an `rq` entry in the `/etc/vfstab` file are checked and assigned correct initial values.

You can run `quotacheck` on individual file systems by typing `quotacheck /dev/dsk/filesystem` and pressing Return. For example, to check quotas for `/files` on slice `/dev/dsk/c0t4d0s2`, type:

```
quotacheck /dev/dsk/c0t4d0s2
```

▼ How to Report on Quotas and Disk Use

To quickly find out if any quotas are being exceeded:

◆ Type `quota` and press Return.

Warnings about mounted file systems where usage is over the quota for the current user are displayed.

To report the quotas and disk use for an individual user on file systems to which quotas apply:

1. Become superuser.

Only privileged users can display information about other users' quotas.

2. Type `quota -v` and press Return.

User quotas on all mounted file systems where quotas are enabled are displayed.

To report on quotas and disk use for all users on one or more file systems:

1. Become superuser.**2. Type `repquota -v /dev/dsk/cntndnsn` and press Return.**

All quotas for the specified file system are displayed, even if there is no usage.

Alternatively, use `repquota -a` to report on all file systems.

To display information about file ownership for a file system:

1. Become superuser.**2. Type `quot /dev/dsk/cntndnsn` and press Return.**

A list of users is displayed with the number of 1024-byte blocks currently owned by each user.

Note – The `quot` command and the `quota` command are two different commands. Use `quot` whether or not quotas are enabled to find out information about which users are accessing disk space in a file system. See the `quot(1M)` manual page for other options.

```
pluto% su
Password:
# quot /dev/dsk/c0t3d0s0
/dev/dsk/c0t3d0s0 (/export/home):
 7596  root
 2402  bin
  202  lp
  178  adm
   47  uucp
   22  sys
    8  jones
    1  svvs
#
```


Part 4 — Troubleshooting

This part has two chapters:

Chapter 11, “Recognizing File Access Problems,” explains how to recognize and repair common problems users encounter when trying to access files.

Chapter 12, “Checking File System Integrity,” explains the `fsck` file system check program and how to use it.

Recognizing File Access Problems

11 

This chapter contains these parts:

<i>Recognizing Problems With Search Paths</i>	<i>page 209</i>
<i>Recognizing Problems With Permission and Ownership</i>	<i>page 212</i>
<i>Recognizing Problems With Network Access</i>	<i>page 215</i>

Users frequently experience problems—and call on a system administrator for help—because they cannot access a program, a file, or a directory that they used to be able to use. Whenever you encounter such a problem, investigate one of three areas:

- The user's search path may have been changed, or the directories in the search path may not be in the proper order.
- The file or directory may not have the proper permissions or ownership.
- The configuration of a system accessed over the network may have changed.

This chapter briefly describes how to recognize problems in each of these three areas and suggests possible solutions.

Recognizing Problems With Search Paths

If a user types a command that is not in the search path, the message `Command not found` is displayed. The command may not be found because:

- The command is not available on the system.

- The command directory is not in the search path.

If the wrong version of the command is found, a directory with a command of the same name is in the search path. In this case, the proper directory may be later in the search path or may not be present at all.

Diagnosing problems with search paths involves these steps:

1. **Displaying the current search path.**
2. **Editing the file where the user's path is set (.login or .cshrc for the C shell, .profile for the Bourne and Korn shells). Add the directory to the path definition, or rearrange the order of the directories listed for the path.**

Note – For the C shell, always check both the .cshrc and .login files to make sure the path information is set all in one place. Duplicate entries can make the search path hard to troubleshoot and make search times less efficient for the user.

3. **Sourcing the file to activate the changes.**
4. **Verifying that the command is found in the right place.**
5. **Executing the command.**

The tasks you use to follow this procedure are described below.

▼ How to Display the Current Search Path

- ♦ **Type `echo $PATH` and press Return.**
The current search path is displayed.

```
mars% echo $PATH
/sbin:/usr/sbin:/usr/bin:/etc
mars%
```

▼ How to Set the Path for Bourne and Korn Shells

The path for the Bourne and Korn shells is specified in the user's `$HOME/.profile` file in this way:

```
PATH=./usr/bin:/$HOME/bin;export PATH
```

▼ How to Set the Path for the C Shell

The path for the C shell is specified in the user's `$HOME/.cshrc` file (with the `set path` environment variable) in this way:

```
set path = ( . /usr/bin $home/bin)
```

▼ How to Source C Shell Dot Files

To source the `.cshrc` file:

♦ **Type `source .cshrc` and press Return.**

New information in the file is made available to the shell.

To source the `.login` file:

♦ **Type `source .login` and press Return.**

New information in the file is made available to the shell.

▼ How to Source Korn and Bourne Shell Dot Files

To source the `.profile` file:

♦ **Type `. .profile` and press Return.**

New information in the file is made available to the shell.

▼ How to Verify the Search Path

♦ **Type which *command-name* and press Return.**

If the command is found in the path, the path and the name of the command are displayed.

Note – The `which` command looks in the `.cshrc` file for information. The `which` command may give misleading results if you execute it from the Bourne or Korn shell and you have a `.cshrc` file that contains aliases for `which`. To ensure accurate results, use the `which` command in a C shell, or, in the Korn shell, use the `whence` command.

This example shows that the OpenWindows executable is not in any of the directories in the search path:

```
venus% which openwin
no openwin in . /home/ignatz /sbin /usr/sbin /usr/bin /etc \
/home/ignatz/bin /bin /home/bin /usr/etc
venus%
```

This example shows that the executable for OpenWindows is found among the directories in the search path:

```
venus% which openwin
/usr/openwin
venus%
```

If you cannot find a command, look at the manual page. For example, if you cannot find the `lpsched` command (the `lp` printer daemon), the `lpsched(1M)` manual page tells you the path is `/usr/lib/lp/lpsched`.

▼ How to Execute a Command

- ◆ Type *command-name* and press **Return**.

Recognizing Problems With Permission and Ownership

When users cannot access files or directories that they used to be able to access, the most likely problem is that permissions or ownership of the files or directories has changed.

Frequently, file and directory ownerships change because someone edited the files as `root`. When you create home directories for new users, be sure to make the user the owner of the dot (`.`) file in the home directory. When users do not own “`.`” they cannot create files in their own home directory.

Access problems can also arise when the group ownership changes or when a group of which a user is a member is deleted from the `/etc/groups` database.

▼ How to Change File Ownership

Note – You must own a file or directory (or have root permission) to be able to change its owner.

1. **Type `ls -l filename` and press Return.**
The owner of the file is displayed in the third column.
2. **Become superuser.**
3. **Type `chown new-owner filename` and press Return.**
Ownership is assigned to the new owner you specify.

```
venus% ls -l quest
-rw-r--r-- 1 fred  staff    6023 Aug  5 12:06 quest
venus% su
Password:
# chown ignatz quest
# ls -l quest
-rw-r--r-- 1 ignatz  staff    6023 Aug  5 12:06 quest
#
```

▼ How to Change File Permissions

Table 11-1 shows the octal values for setting file permissions. You use these numbers in sets of three to set permissions for owner, group, and other. For

example, the value 644 sets read/write permissions for owner, and read-only permissions for group and other.

Table 11-1 Octal Values for File Permissions

Value	Description
0	No permissions
1	Execute-only
2	Write-only
3	Write, execute
4	Read-only
5	Read, execute
6	Read, write
7	Read, write, execute

1. Type `ls -l filename` and press Return.

The long listing shows the current permissions for the file.

2. Type `chmod nnn filename` and press Return.

Permissions are changed using the numbers you specify.

Note - You can change permissions on groups of files or on all files in a directory using meta characters such as (*?) in place of file names or in combination with them.

This example shows changing the permissions of a file from 666 (read/write, read/write, read/write) to 644 (read/write, read-only, read-only).

```
venus% ls -l quest
-rw-rw-rw- 1 ignatz  staff    6023 Aug  5 12:06 quest
venus% chmod 644 quest
venus% ls -l
-rw-r--r-- 1 ignatz  staff    6023 Aug  5 12:06 quest
venus%
```


▼ How to Change File Group Ownership

◆ **Type `chgrp gid filename` and press Return.**

The group ID for the file you specify is changed.

```
$ ls -lg junk
-rw-r--r-- 1 other 3138 Oct 31 14:49 junk
$ chgrp 10 junk
$ ls -lg junk
-rw-r--r-- 1 staff  3138 Oct 31 14:49 junk
$
```

See *User Accounts, Printers, and Mail Administration* and *Name Services Administration Guide* for information about how to edit group accounts.

Recognizing Problems With Network Access

If users have problems using the `rmp` remote copy command to copy files over the network, the directories and files on the remote system may have restricted access by setting permissions. Another possible source of trouble is that the remote system and the local system are not configured to allow access.

See *Security, Performance, and Accounting Administration* and *NFS Administration Guide* for information about problems with network access and problems with accessing systems through `autofs`.

Checking File System Integrity

12 

This chapter contains these sections:

<i>Understanding How the File System State Is Recorded</i>	<i>page 218</i>
<i>What fsck Checks and Tries to Repair</i>	<i>page 219</i>
<i>Error Messages</i>	<i>page 226</i>
<i>Modifying Automatic Boot Checking</i>	<i>page 253</i>
<i>Interactively Checking and Repairing a UFS File System</i>	<i>page 255</i>
<i>Restoring a Bad Superblock</i>	<i>page 258</i>
<i>Syntax and Options for the fsck Command</i>	<i>page 260</i>

The UFS file system relies on an internal set of tables to keep track of inodes and used and available blocks. When these internal tables are not properly synchronized with data on a disk, inconsistencies result and file systems need to be repaired.

File systems can be damaged or become inconsistent because of abrupt termination of the operating system in these ways:

- Power failure
- Accidental unplugging of the system
- Turning the system off without proper shutdown procedure
- A software error in the kernel

File system corruption, while serious, is not common. When a system is booted, a file system consistency check is automatically done. Most of the time, this file system check repairs problems it encounters.

File systems are checked with the `fsck` (file system check) program. This chapter describes the flags to `fsck`, what `fsck` checks and repairs, and `fsck` error messages. It also describes how to modify the automatic checking done during booting, how to find out if a file system needs to be checked, how to check and repair a UFS file system interactively, how to restore a bad superblock, and how to fix a UFS file system that `fsck` cannot repair.

The `fsck` command puts files and directories that are allocated but unreferenced in the `lost+found` directory. The inode number of each file is assigned as the name. If the `lost+found` directory does not exist, `fsck` creates it. If there is not enough space in the `lost+found` directory, `fsck` increases its size.

Understanding How the File System State Is Recorded

The Solaris 2.3 `fsck` command uses a *state flag*, which is stored in the superblock, to record the condition of the file system. This flag is used by the `fsck` command to determine whether or not a file system needs to be checked for consistency. The flag is used by the `/etc/bcheckrc` script during booting and by the `fsck` command when run from a command line using the `-m` option. If you ignore the result from the `-m` option to `fsck`, all file systems can be checked regardless of the setting of the state flag.

The possible state values are:

- `FSCLEAN` – If the file system was unmounted properly, the state flag is set to `FSCLEAN`. Any file system with an `FSCLEAN` state flag is not checked when the system is booted.
- `FSSTABLE` – The file system is (or was) mounted but has not changed since the last checkpoint: `sync` or `fsflush`—which normally occurs every 30 seconds. For example, the kernel periodically checks to see if a file system is idle and, if so, flushes the information in the superblock back to the disk and marks it `FSSTABLE`. If the system crashes, the file system structure is stable, but users may lose a small amount of data. File systems that are marked `FSSTABLE` can skip the checking before mounting.
- `FSACTIVE` – When a file system is mounted and then modified, the state flag is set to `FSACTIVE`. The file system may contain inconsistencies. A file system will be marked as `FSACTIVE` before any modified metadata is written to the disk. When a file system is unmounted gracefully, the state flag is set to `FSCLEAN`. A file system with the `FSACTIVE` flag must be

checked by `fsck` because it may be inconsistent. The `mount(2)` system call will not mount a file system for read/write if the file system state is not `FSCLEAN` or `FSSTABLE`.

- `FSBAD` – When the root file system is mounted when its state is not `FSCLEAN` or `FSSTABLE`, the state flag is set to `FSBAD`. The kernel will not change this file system state to `FSCLEAN` or `FSSTABLE`. If a root file system is flagged `FSBAD` as part of the boot process, it will be mounted read-only. You can run `fsck` on the raw root device. Then remount the root file system as read/write.

Table 12-1 shows when the state flag is modified.

Table 12-1 State Flag Transitions After `fsck`

Before <code>fsck</code>	After <code>fsck</code>		
	No Errors	New State All Errors Corrected	Uncorrected Errors
unknown	stable	stable	unknown
active	stable	stable	active
stable	stable	stable	active
clean	clean	stable	active
bad	stable	stable	bad

What `fsck` Checks and Tries to Repair

This section describes what happens in the normal operation of a file system, what can go wrong, what problems `fsck` (the checking and repair utility) looks for, and how it corrects the inconsistencies it finds.

Why Inconsistencies May Occur

Every working day hundreds of files may be created, modified, and removed. Each time a file is modified, the operating system performs a series of file system updates. These updates, when written to the disk reliably, yield a consistent file system.

When a user program does an operation to change the file system, such as a *write*, the data to be written is first copied into an internal in-core buffer in the kernel. Normally, the disk update is handled asynchronously; the user process

is allowed to proceed even though the data *write* may not happen until long after the `write` system call has returned. Thus at any given time, the file system, as it resides on the disk, lags behind the state of the file system represented by the in-core information.

The disk information is updated to reflect the in-core information when the buffer is required for another use or when the kernel automatically runs the `fsflush` daemon (at 30-second intervals). If the system is halted without writing out the in-core information, the file system on the disk will be in an inconsistent state.

A file system can develop inconsistencies in several ways. The most common causes are operator error and hardware failures.

Problems may result from an *unclean halt*, if a system is shut down improperly, or when a mounted file system is taken offline improperly. To prevent unclean halts, the current state of the file systems must be written to disk (that is, “synchronized”) before halting the CPU, physically taking a disk pack out of a drive, or taking a disk offline.

Inconsistencies can also result from defective hardware. Blocks can become damaged on a disk drive at any time, or a disk controller can stop functioning correctly.

The UFS Components That Are Checked for Consistency

This section describes the kinds of consistency checks applied to these UFS file system components: superblock, cylinder group blocks, inodes, indirect blocks, and data blocks.

Superblock

The superblock stores summary information, which is the most commonly corrupted item in a UFS file system. Each change to the file system data blocks or inodes also modifies the superblock. If the CPU is halted and the last command is not a `sync` command, the superblock will almost certainly be corrupted.

The superblock is checked for inconsistencies in:

- File system size
- Number of inodes

- Free-block count
- Free-inode count

File System and Inode List Size

The file system size must be larger than the number of blocks used by the superblock and the number of blocks used by the list of inodes. The number of inodes must be less than the maximum number allowed for the file system. The file system size and layout information are the most critical pieces of information for `fsck`. Although there is no way to actually check these sizes, because they are statically determined when the file system is created, `fsck` can check that the sizes 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 primary superblock, it requests the operator to specify the location of an alternate superblock.

Free Blocks

Free blocks are stored in the cylinder group block maps. `fsck` checks that all the blocks marked as free are not claimed by any files. When all the blocks have been accounted for, a check is made to see if 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, leaving out blocks already allocated.

The summary information in the superblock contains a count of the total number of free blocks within the file system. The `fsck` program compares this count to the number of free blocks it finds within the file system. If the counts do not agree, `fsck` replaces the count in the superblock with the actual free-block count.

Free Inodes

The summary information in the superblock contains a count of the number of free inodes within the file system. The `fsck` program compares this count to the number of free inodes it finds within the file system. If the counts do not agree, `fsck` replaces the count in the superblock with the actual free inode count.

Inodes

The list of inodes is checked sequentially starting with inode 2 (inode 0 and inode 1 are reserved). Each inode is checked for inconsistencies in:

- Format and type
- Link count
- Duplicate block
- Bad block numbers
- Inode size

Format and Type of Inodes

Each inode contains a mode word, which 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:

- Allocated
- Unallocated
- Partially allocated

When the file system is created, a fixed number of inodes are set aside, but they are not allocated until they are needed. An allocated inode is one that points to a file. An unallocated inode does not point to a file and, therefore, should be empty. The partially allocated state means that the inode is incorrectly formatted. An inode can get into this state if, for example, bad data is written into the inode list because of a hardware failure. The only corrective action `fsck` can take is to clear the inode.

Link Count

Each inode contains a count of the number of directory entries linked to it. The `fsck` program 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 and the actual count is 0.

This condition can occur if no directory entry exists for the inode. In this case, `fsck` puts the disconnected file in the `lost+found` directory.

- The stored count is *not* 0 and the actual count is *not* 0, but the counts are *unequal*.

This condition can occur if a directory entry has been added or removed but the inode has not been updated. In this case, `fsck` replaces the stored link count with the actual link count.

- The stored count is 0 and the actual count is not 0.

In this case `fsck` changes the link count of the inode to the actual count.

Duplicate Blocks

Each inode contains a list, or pointers to lists (indirect blocks), of all the blocks claimed by the inode. Because indirect blocks are owned by an inode, inconsistencies in indirect blocks directly affect the inode that owns the indirect block.

The `fsck` program compares each block number claimed by an inode to a list of allocated blocks. If another inode already claims a block number, the block number is put on 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` 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.) It is not possible to determine with certainty which inode is in error. The `fsck` program prompts you to choose which inode should be kept and which should be cleared.

Bad Block Numbers

The `fsck` program 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 data 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. The `fsck` program prompts you to clear the inode.

Inode Size

Each inode contains a count of the number of data blocks that it references. The number of actual data blocks is the sum of the allocated data blocks and the indirect blocks. `fsck` computes the number of data blocks and compares that block count against the number of blocks the inode claims. If an inode contains an incorrect count, `fsck` prompts you to fix it.

Each inode contains a 64-bit size field. This field shows the number of characters (data bytes) in the file associated with the inode. A rough check of the consistency of the size field of an inode is done by using the number of characters shown in the size field to calculate how many blocks should be associated with the inode, and then comparing 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 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 are also performed for indirect blocks.

Directory Data Blocks

An inode can directly or indirectly reference three kinds of data blocks. All referenced blocks must be of 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 symbolic link. Directory data blocks contain directory entries. `fsck` can check the validity only of 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. Directory data blocks are checked 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 the inode number in a directory data block points to an unallocated inode, `fsck` removes the directory entry. This condition can occur if the data blocks containing the directory entries are modified and written out but the inode does not get written out. This condition can occur if the CPU is halted without warning.

Bad Inode Number

If a directory entry inode number points beyond the end of the inode list, `fsck` removes the directory entry. This condition can occur when 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. It must reference itself; that is, its value must be equal to 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 be equal to the inode number of the parent directory (or the inode number of itself if the directory is the root directory).

If the directory inode numbers for “.” and “..” are incorrect, `fsck` replaces them with the correct values. If there are multiple hard links to a directory, the first one found is considered the real parent to which “..” should point. In this case, `fsck` recommends you have it delete the other names.

Disconnected Directories

The `fsck` program checks the general connectivity of the file system. If a directory is found 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.)

Regular Data Blocks

Data blocks associated with a regular file hold the contents of the file. `fsck` does not attempt to check the validity of the contents of a regular file's data blocks.

Error Messages

Normally, `fsck` is run non-interactively to *preen* the file systems after an abrupt system halt in which the latest file system changes were not written to disk. Preening automatically fixes any basic file system inconsistencies and does not try to repair more serious errors. While preening a file system, `fsck` fixes the inconsistencies it expects from such an abrupt halt. For more serious conditions, the command reports the error and terminates.

When you run `fsck` interactively, `fsck` reports each inconsistency found and fixes innocuous errors. However, for more serious errors, the command reports the inconsistency and prompts you to choose a response. When you run `fsck` using the `-y` or `-n` options, your response is predefined as yes or no to the default response suggested by `fsck` for each error condition.

Some corrective actions will result in some loss of data. The amount and severity of data loss may be determined from the `fsck` diagnostic output.

`fsck` is a multipass file system check program. Each pass invokes a different phase of the `fsck` program with different sets of messages. 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.

The phases (passes) performed by the UFS version of `fsck` are:

- Initialization
- Phase 1 – Check blocks and sizes

- Phase 2 – Check path names
- Phase 3 – Check connectivity
- Phase 4 – Check reference counts
- Phase 5 – Check cylinder groups

The next sections describe the error conditions that may be detected in each phase, the messages and prompts that result, and possible responses you can make.

Messages that may appear in more than one phase are described in “General fsck Error Messages” on page 227. Otherwise, messages are organized by the phases in which they occur. In addition, to help you look up a message, they are presented in alphabetical order.

Many of the messages include the abbreviations shown in Table 12-2:

Table 12-2 Abbreviations Used in fsck Messages

Abbreviation	Meaning
BLK	Block number
DUP	Duplicate block number
DIR	Directory name
CG	Cylinder group
MTIME	Time file was last modified
UNREF	Unreferenced

Many of the messages also include variable fields, such as inode numbers, which are represented in this book by an italicized term, such as *inode-number*. For example, this screen message:

```
INCORRECT BLOCK COUNT I=2529
```

is shown as:

```
INCORRECT BLOCK COUNT I=inode-number
```

General fsck Error Messages

The error messages in this section may be displayed in any phase after initialization. Although they offer the option to continue, it is generally best to regard them as fatal. They reflect a serious system failure and should be

handled immediately. When confronted with such a message, terminate the program with a no response. If you cannot determine what caused the problem, contact your local service provider or another qualified person.

```
CANNOT SEEK: BLK block-number (CONTINUE)
```

A request to move to a specified block number *block-number* in the file system failed. This message indicates a serious problem, probably a hardware failure.

If you want to continue the file system check, do a second run of `fsck` to recheck the file system. If the block was part of the virtual memory buffer cache, `fsck` will terminate with:

```
Fatal I/O error
```

```
CANNOT READ: BLK block-number (CONTINUE)
```

A request to read a specified block number *block-number* in the file system failed. The message indicates a serious problem, probably a hardware failure. If you want to continue the file system check, `fsck` will retry the read and display this message:

```
THE FOLLOWING SECTORS COULD NOT BE READ: sector-numbers
```

where *sector-numbers* indicates the sectors that could not be read. If `fsck` tries to write back one of the blocks on which the read failed it displays this message:

```
WRITING ZERO'ED BLOCK sector-numbers TO DISK
```

where *sector-numbers* indicates the sector that was written with zeroes. If the disk is experiencing hardware problems, the problem will persist. This error condition prevents a complete check of the file system. Run `fsck` again to recheck the file system. If the block was part of the virtual memory buffer cache, `fsck` terminates and displays this error message:

```
Fatal I/O error
```

CANNOT WRITE: BLK *block-number* (CONTINUE)

A request to write a specified block number *block-number* in the file system failed. The disk may be write-protected. Check the write-protect lock on the drive. If that is not the problem, contact your local service provider or another qualified person. If you continue the file system check, the write operation will be retried. Sectors that could not be written are shown with this message:

THE FOLLOWING SECTORS COULD NOT BE WRITTEN: *sector-numbers*

where *sector-numbers* indicates the sectors that could not be written. If the disk has 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 was part of the virtual memory buffer cache, `fsck` terminates and displays this error message:

Fatal I/O error

Initialization Phase fsck Messages

In the initialization phase, command-line syntax is checked. Before the file system check can be performed, `fsck` sets up tables and opens 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. All such initialization errors terminate `fsck` when it is preening the file system.

bad inode number *inode-number* to ginode

An internal error occurred because of a nonexistent inode *inode-number*. `fsck` exits. If this message is displayed, contact your local service provider or another qualified person.

cannot alloc *size-of-block map* bytes for blockmap
cannot alloc *size-of-free map* bytes for freemap
cannot alloc *size-of-state map* bytes for statemap
cannot alloc *size-of-lncntp* bytes for lncntp

Request for memory for its internal tables failed. `fsck` terminates. This message indicates a serious system failure that should be handled immediately. This condition may occur if other processes are using a very large amount of system resources. Killing other processes may solve the problem. Contact your local service provider or another qualified person.

Can't open checklist file: *filename*

The file system checklist file *filename* (usually `/etc/vfstab`) cannot be opened for reading. `fsck` terminates. Check to see if the file exists and if its access modes permit read access.

Can't open *filename*

`fsck` cannot open file system *filename*. When running interactively, `fsck` ignores this file system and continues checking the next file system given. Check to see if read and write access to the raw device file for the file system is permitted.

Can't stat root

`fsck` request for statistics about the root directory failed. `fsck` terminates. This message indicates a serious system failure. Contact your local service provider or another qualified person.

Can't stat *filename*

Can't make sense out of name *filename*

`fsck` request for statistics about the file system *filename* failed. When running interactively, `fsck` ignores this file system and continues checking the next file system given. Check to see if the file system exists and check its access modes.

filename: (NO WRITE)

Either the `-n` option was specified or `fsck` could not open the file system *filename* for writing. When `fsck` is running in no-write mode, all diagnostic messages are displayed, but `fsck` does not attempt to fix anything.

If `-n` was not specified, check the type of the file specified. It may be the name of a regular file.

IMPOSSIBLE MINFREE=*percent* IN SUPERBLOCK (SET TO DEFAULT)

The superblock minimum space percentage is greater than 99 percent or less than 0 percent.

At the SET TO DEFAULT prompt, type:

y - To set the minfree parameter to 10 percent.

n - To ignore this error condition.

INTERNAL INCONSISTENCY: *message*

`fsck` has had an internal error, whose message is *message*. If this message is displayed, contact your local service provider or another qualified person.

MAGIC NUMBER WRONG

NCG OUT OF RANGE

CPG OUT OF RANGE

NCYL DOES NOT JIBE WITH NCG*CPG

SIZE PREPOSTEROUSLY LARGE

TRASHED VALUES IN SUPER BLOCK

followed by

filename: BAD SUPER BLOCK: *block-number*

USE AN ALTERNATE SUPER-BLOCK TO SUPPLY NEEDED INFORMATION;

e.g., `fsck[-f ufs] -o b=# [special ...]`

where # is the alternate superblock. See `fsck_ufs(1M)`

The superblock has been corrupted. Use an alternative superblock to supply needed information. Specifying block 32 is a good first choice. You can locate an alternative copy of the superblock by running the `newfs -N` command on the slice.



Caution – Be sure to specify the `-N` option; otherwise, `newfs` overwrites the existing file system.

UNDEFINED OPTIMIZATION IN SUPERBLOCK (SET TO DEFAULT)

The superbblock optimization parameter is neither `OPT_TIME` nor `OPT_SPACE`.

At the `SET TO DEFAULT` prompt, type:

`y` – To minimize the time to perform operations on the file system.

`n` – To ignore this error condition.

Phase 1: Check Blocks and Sizes Messages

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 errors in this phase except `INCORRECT BLOCK COUNT`, `PARTIALLY TRUNCATED INODE`, `PARTIALLY ALLOCATED INODE`, and `UNKNOWN FILE TYPE` terminate `fsck` when it is preening a file system.

These messages (in alphabetical order) may occur in phase 1:

block-number BAD I=*inode-number*

Inode *inode-number* contains a block number *block-number* 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 *inode-number* has too many block numbers outside the file system range. This error condition generates the `BAD/DUP` error message in phases 2 and 4.

BAD MODE: MAKE IT A FILE?

The status of a given inode is set to all 1s, indicating file system damage. This message does not indicate physical disk damage, unless it is displayed repeatedly after `fsck -y` has been run. Type `y` to reinitialize the inode to a reasonable value.

BAD STATE *state-number* TO BLKERR

An internal error has scrambled the `fsck` state map so that it shows the impossible value *state-number*. `fsck` exits immediately. If this error message is displayed, contact your local service provider or another qualified person.

block-number DUP I=*inode-number*

Inode *inode-number* contains a block number *block-number*, which is already claimed by the same or another inode. This error condition may generate the EXCESSIVE DUP BLKS error message in phase 1 if inode *inode-number* has too many block numbers claimed by the same or another inode. This error condition invokes phase 1B and generates the BAD/DUP error messages in phases 2 and 4.

DUP TABLE OVERFLOW (CONTINUE)

There is no more room in an internal table in `fsck` containing duplicate block numbers. If the `-o p` option is specified, the program terminates.

Type:

`y` – To continue the program. When this error occurs, a complete check of the file system is not possible. If another duplicate block is found, this error condition repeats. Increase the amount of virtual memory available (by killing some processes, increasing swap space) and run `fsck` again to recheck the file system.

`n` – To terminate the program.

EXCESSIVE BAD BLOCKS I=*inode-number* (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 *inode-number*. If the `-o p` (preen) option is specified, the program terminates.

Type:

y - To continue the program. When this error occurs, a complete check of the file system is not possible. You should run `fsck` again to recheck the file system.

n - To terminate the program.

EXCESSIVE DUP BLKS I=*inode-number* (CONTINUE)

Too many (usually more than 10) blocks are claimed by the same or another inode or by a free-list. If the `-o p` option is specified, the program terminates.

Type:

y - To continue the program. When this error occurs, a complete check of the file system is not possible. A second run of `fsck` should be made to recheck the file system.

n - To terminate the program.

INCORRECT BLOCK COUNT I=*inode-number* (*number-of-BAD-DUP-or-missing-blocks* should be *number-of-blocks-in-filesystem*) (CORRECT)

The block count for inode *inode-number* is *number-of-BAD-DUP-or-missing-blocks*, but should be *number-of-blocks-in-filesystem*. When preening, `fsck` corrects the count.

At the CORRECT prompt, type:

y - To replace the block count of inode *inode-number* by *number-of-blocks-in-filesystem*.

n - To ignore this error condition.

LINK COUNT TABLE OVERFLOW (CONTINUE)

There is no more room in an internal table for `fsck` containing allocated inodes with a link count of zero. If the `-o p` (preen) option is specified, the program exits and `fsck` has to be completed manually.

Type:

`y` - To continue the program. If another allocated inode with a zero-link count is found, this error condition repeats. When this error occurs, a complete check of the file system is not possible. A second run of `fsck` should be made to recheck the file system. Increase the virtual memory available by killing some processes, or increasing swap space, before running `fsck` again.

`n` - To terminate the program.

PARTIALLY ALLOCATED INODE I=*inode-number* (CLEAR)

Inode *inode-number* is neither allocated nor unallocated. If the `-o p` (preen) option is specified, the inode is cleared.

Type:

`y` - To deallocate the inode *inode-number* by zeroing out its contents. This may generate the UNALLOCATED error condition in phase 2 for each directory entry pointing to this inode.

`n` - To ignore the error condition. A no response is appropriate only if you intend to take other measures to fix the problem.

PARTIALLY TRUNCATED INODE I=*inode-number* (SALVAGE)

`fsck` has found inode *inode-number* whose size is shorter than the number of blocks allocated to it. This condition occurs only if the system crashes while truncating a file. When preening the file system, `fsck` completes the truncation to the specified size.

At the SALVAGE prompt, type:

`y` - To complete the truncation to the size specified in the inode.

`n` - To ignore this error condition.

UNKNOWN FILE TYPE I=*inode-number* (CLEAR)

The mode word of the inode *inode-number* shows that the inode is not a pipe, special character inode, special block inode, regular inode, symbolic link, FIFO file, or directory inode. If the `-o p` option is specified, the inode is cleared.

Type:

`y` - To deallocate the inode *inode-number* by zeroing its contents, which results in the `UNALLOCATED` error condition in phase 2 for each directory entry pointing to this inode.

`n` - To ignore this error condition.

Phase 1B: Rescan for More DUPS Messages

When a duplicate block is found in the file system, this informational message is displayed:

block-number DUP I=*inode-number*

Inode *inode-number* contains a block number *block-number* that is already claimed by the same or another inode. This error condition generates the `BAD/DUP` error message in phase 2. Inodes that have overlapping blocks may be determined by examining this error condition and the `DUP` error condition in phase 1.

When a duplicate block is found, the file system is rescanned to find the inode that previously claimed that block.

Phase 2: Check Path Names

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

When the file system is being preened (`-o p` option), all errors in this phase terminate `fsck`, except those related to directories not being a multiple of the block size, duplicate and bad blocks, inodes out of range, and extraneous hard links.

BAD INODE *state-number* TO DESCEND

An `fsck` internal error has passed an impossible state *state-number* to the routine that descends the file system directory structure. `fsck` exits. If this error message is displayed, contact your local service provider or another qualified person.

BAD INODE NUMBER FOR '.' I=*inode-number* OWNER=*UID* MODE=*file-mode* SIZE=*file-size* MTIME=*modification-time* DIR=*filename* (FIX)

A directory *inode-number* has been found whose inode number for “.” does not equal *inode-number*.

At the FIX prompt, type:

y - To change the inode number for “.” to be equal to *inode-number*.

n - To leave the inode number for “.” unchanged.

BAD INODE NUMBER FOR '..' I=*inode-number* OWNER=*UID* MODE=*file-mode* SIZE=*file-size* MTIME=*modification-time* DIR=*filename* (FIX)

A directory *inode-number* has been found whose inode number for “..” does not equal the parent of *inode-number*.

At the FIX prompt, type:

y - To change the inode number for “..” to be equal to the parent of *inode-number*. (Note that “..” in the root inode points to itself.)

n - To leave the inode number for “..” unchanged.

BAD RETURN STATE *state-number* FROM DESCEND

An `fsck` internal error has returned an impossible state *state-number* from the routine that descends the file system directory structure. `fsck` exits. If this message is displayed, contact your local service provider or another qualified person.

BAD STATE *state-number* FOR ROOT INODE

An internal error has assigned an impossible state *state-number* to the root inode. `fsck` exits. If this error message is displayed, contact your local service provider or another qualified person.

BAD STATE *state-number* FOR INODE=*inode-number*

An internal error has assigned an impossible state *state-number* to inode *inode-number*. `fsck` exits. If this error message is displayed, contact your local service provider or another qualified person.

DIRECTORY TOO SHORT I=*inode-number* OWNER=*UID* MODE=*file-mode*
SIZE=*file-size* MTIME=*modification-time* DIR=*filename* (FIX)

A directory *filename* has been found whose size *file-size* is less than the minimum directory size. The owner *UID*, mode *file-mode*, size *file-size*, modify time *modification-time*, and directory name *filename* are displayed.

At the FIX prompt, type:

- y - To increase the size of the directory to the minimum directory size.
- n - To ignore this directory.

DIRECTORY *filename*: LENGTH *file-size* NOT MULTIPLE OF *block-number*
(ADJUST)

A directory *filename* has been found with size *file-size* that is not a multiple of the directory block size *block-number*.

At the ADJUST prompt, type:

- y - To round up the length to the appropriate block size. When preening the file system (`-o p` option), `fsck` only displays a warning and adjusts the directory.
- n - To ignore this condition.

DIRECTORY CORRUPTED I=*inode-number* OWNER=*UID* MODE=*file-mode*
SIZE=*file-size* MTIME=*modification-time* DIR=*filename* (SALVAGE)

A directory with an inconsistent internal state has been found.

At the SALVAGE prompt, type:

y - To 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. Take this action only after other recovery efforts have failed.

n - To skip to the next directory boundary and resume reading, but not modify the directory.

DUP/BAD I=*inode-number* OWNER=0 MODE=M SIZE=*file-size*
MTIME=*modification-time* TYPE=*filename* (REMOVE)

Phase 1 or phase 1B found duplicate blocks or bad blocks associated with directory or file entry *filename*, inode *inode-number*. The owner *UID*, mode *file-mode*, size *file-size*, modify time *modification-time*, and directory or file name *filename* are displayed. If the -p (preen) option is specified, the duplicate/bad blocks are removed.

At the SALVAGE prompt, type:

y - To remove the directory or file entry *filename*.

n - To ignore this error condition.

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.

At the REALLOCATE prompt, type:

y - To clear the existing contents of the root inode and reallocate it. The files and directories usually found in the root will be recovered in phase 3 and put into the `lost+found` directory. If the attempt to allocate the root fails, `fsck` will exit with: CANNOT ALLOCATE ROOT INODE.

n - To get the CONTINUE prompt. To respond to the CONTINUE prompt, type:

y - To ignore the DUPS/BAD error condition in the root inode and continue running the file system check. If the root inode is not correct, this may generate many other error messages.

n - To terminate the program.

```
EXTRA '.' ENTRY I=inode-number OWNER=UID MODE=file-mode SIZE=file-size MTIME=modification-time DIR=filename (FIX)
```

A directory *inode-number* has been found that has more than one entry for “.”.

At the FIX prompt, type:

y - To remove the extra entry for “.”

n - To leave the directory unchanged.

```
EXTRA '..' ENTRY I=inode-number OWNER=UID MODE=file-mode SIZE=file-size MTIME=modification-time DIR=filename (FIX)
```

A directory *inode-number* has been found that has more than one entry for “..” (the parent directory).

At the FIX prompt, type:

y - To remove the extra entry for “..” (the parent directory).

n - To leave the directory unchanged.

```
hard-link-number IS AN EXTRANEOUS HARD LINK TO A DIRECTORY filename (REMOVE)
```

fsck has found an extraneous hard link *hard-link-number* to a directory *filename*. When preening (-o p option), *fsck* ignores the extraneous hard links.

At the SALVAGE prompt, type:

y - To delete the extraneous entry *hard-link-number*.

n - To ignore the error condition.

inode-number OUT OF RANGE I=*inode-number* NAME=*filename* (REMOVE)

A directory entry *filename* has an inode number *inode-number* that is greater than the end of the inode list. If the `-p` (preen) option is specified, the inode will be removed automatically.

At the SALVAGE prompt, type:

y - To delete the directory entry *filename*.

n - To ignore the error condition.

MISSING '.' I=*inode-number* OWNER=*UID* MODE=*file-mode* SIZE=*file-size*
MTIME=*modification-time* DIR=*filename* (FIX)

A directory *inode-number* has been found whose first entry (the entry for ".") is unallocated.

At the FIX prompt, type:

y - To build an entry for "." with inode number equal to *inode-number*.

n - To leave the directory unchanged.

MISSING '.' I=*inode-number* OWNER=*UID* MODE=*file-mode* SIZE=*file-size*
MTIME=*modification-time* DIR=*filename*
CANNOT FIX, FIRST ENTRY IN DIRECTORY CONTAINS *filename*

A directory *inode-number* has been found whose first entry is *filename*. `fsck` cannot resolve this problem. Mount the file system and move entry *filename* elsewhere. Unmount the file system and run `fsck` again.

MISSING '.' I=*inode-number* OWNER=*UID* MODE=*file-mode* SIZE=*file-size*
MTIME=*modification-time* DIR=*filename*
CANNOT FIX, INSUFFICIENT SPACE TO ADD '.'

A directory *inode-number* has been found whose first entry is not ".". `fsck` cannot resolve the problem. If this error message is displayed, contact your local service provider or another qualified person.

```
MISSING '..' I=inode-number OWNER=UID MODE=file-mode SIZE=file-size
MTIME=modification-time DIR=filename (FIX)
```

A directory *inode-number* has been found whose second entry is unallocated.

At the `FIX` prompt, type:

`y` – To build an entry for “`..`” with inode number equal to the parent of *inode-number*. (Note that “`..`” in the root inode points to itself.)

`n` – To leave the directory unchanged.

```
MISSING '..' I=inode-number OWNER=UID MODE=file-mode SIZE=file-size
MTIME=modification-time DIR=filename
CANNOT FIX, SECOND ENTRY IN DIRECTORY CONTAINS filename
```

A directory *inode-number* has been found whose second entry is *filename*. `fsck` cannot resolve this problem. Mount the file system and move entry *filename* elsewhere. Then unmount the file system and run `fsck` again.

```
MISSING '..' I=inode-number OWNER=UID MODE=file-mode SIZE=file-size
MTIME=modification-time DIR=filename
CANNOT FIX, INSUFFICIENT SPACE TO ADD '..'
```

A directory *inode-number* 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.

```
NAME TOO LONG filename
```

An excessively long path name has been found, which usually indicates loops in the file system name space. This error can occur if a privileged user has made circular links to directories. You must remove these links.

ROOT INODE UNALLOCATED (ALLOCATE)

The root inode (usually inode number 2) has no allocate mode bits.

At the ALLOCATE prompt, type:

y - To allocate inode 2 as the root inode. The files and directories usually found in the root will be recovered in phase 3 and put into the `lost+found` directory. If the attempt to allocate the root fails, `fsck` displays this message and exits:

```
CANNOT ALLOCATE ROOT INODE
```

n - To terminate the program.

ROOT INODE NOT DIRECTORY (REALLOCATE)

The root inode (usually inode number 2) of the file system is not a directory inode.

At the REALLOCATE prompt, type:

y - To clear the existing contents of the root inode and reallocate it. The files and directories usually found in the root will be recovered in phase 3 and put into the `lost+found` directory. If the attempt to allocate the root fails, `fsck` displays this message and exits:

```
CANNOT ALLOCATE ROOT INODE
```

n - To have `fsck` prompt with `FIX`.

At the `FIX` prompt, type:

y - To change the type of the root inode to directory. If the root inode's data blocks are not directory blocks, many error messages will be generated.

n - To terminate the program.

UNALLOCATED I=*inode-number* OWNER=*UID* MODE=*file-mode* SIZE=*file-size*
MTIME=*modification-time* type=*filename* (REMOVE)

A directory or file entry *filename* points to an unallocated inode *inode-number*. The owner *UID*, mode *file-mode*, size *file-size*, modify time *modification-time*, and file name *filename* are displayed.

At the SALVAGE prompt, type:

y - To delete the directory entry *filename*.

n - To ignore the error condition.

ZERO LENGTH DIRECTORY I=*inode-number* OWNER=*UID* MODE=*file-mode*
SIZE=*file-size* MTIME=*modification-time* DIR=*filename* (REMOVE)

A directory entry *filename* has a size *file-size* that is zero. The owner *UID*, mode *file-mode*, size *file-size*, modify time *modification-time*, and directory name *filename* are displayed.

At the SALVAGE prompt, type:

y - To remove the directory entry *filename*; this results in the BAD/DUP error message in phase 4.

n - To ignore the error condition.

Phase 3: Check Connectivity Messages

This phase checks the directories examined in phase 2 and reports error conditions resulting from:

- Unreferenced directories
- Missing or full `lost+found` directories

BAD INODE *state-number* TO DESCEND

An internal error has caused an impossible state *state-number* to be passed to the routine that descends the file system directory structure. `fsck` exits. If this occurs, contact your local service provider or another qualified person.

DIR I=*inode-number1* CONNECTED. PARENT WAS I=*inode-number2*

This is an advisory message indicating a directory inode *inode-number1* was successfully connected to the `lost+found` directory. The parent inode *inode-number2* of the directory inode *inode-number1* is replaced by the inode number of the `lost+found` directory.

DIRECTORY *filename* LENGTH *file-size* NOT MULTIPLE OF *block-number*
(ADJUST)

A directory *filename* has been found with size *file-size* that is not a multiple of the directory block size B. (This condition can recur in phase 3 if it is not adjusted in phase 2.)

At the ADJUST prompt, type:

y - To round up the length to the appropriate block size. When preening, `fsck` displays a warning and adjusts the directory.

n - To ignore this error condition.

`lost+found` IS NOT A DIRECTORY (REALLOCATE)

The entry for `lost+found` is not a directory.

At the REALLOCATE prompt, type:

y - To 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 and it will either be reclaimed as an unreferenced inode or have its link count adjusted later in this phase. Inability to create a `lost+found` directory displays the message:

SORRY. CANNOT CREATE `lost+found` DIRECTORY

and aborts the attempt to link up the lost inode, which generates the UNREF error message in phase 4.

n - To abort the attempt to link up the lost inode, which 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.

At the `CREATE` prompt, type:

`y` - To create a `lost+found` directory in the root of the file system. This may lead to the message `NO SPACE LEFT IN / (EXPAND)`. See the explanation of this message below. If the `lost+found` directory cannot be created, `fsck` displays 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 later in phase 4.

`n` - To abort the attempt to link up the lost inode. This generates the `UNREF` error message later in phase 4.

NO SPACE LEFT IN /lost+found (EXPAND)

Another entry cannot be added to the `lost+found` directory in the root directory of the file system because no space is available. When preening, `fsck` expands the `lost+found` directory.

At the `EXPAND` prompt, type:

`y` - To expand the `lost+found` directory to make room for the new entry. If the attempted expansion fails, `fsck` displays:

```
SORRY. NO SPACE IN lost+found DIRECTORY
```

and aborts the request to link a file to the `lost+found` directory. This error generates the `UNREF` error message later in phase 4. Delete any unnecessary entries in the `lost+found` directory. This error terminates `fsck` when preening is in effect.

`n` - To abort the attempt to link up the lost inode. This generates the `UNREF` error message in phase 4.


```
UNREF DIR I=inode-number OWNER=UID MODE=file-mode SIZE=file-size  
MTIME=modification-time (RECONNECT)
```

The directory inode *inode-number* was not connected to a directory entry when the file system was traversed. The owner *UID*, mode *file-mode*, size *file-size*, and modify time *modification-time* of directory inode *inode-number* are displayed. When preening, `fsck` reconnects the non-empty directory inode if the directory size is non-zero. Otherwise, `fsck` clears the directory inode.

At the RECONNECT prompt, type:

`y` – To reconnect the directory inode *inode-number* into the `lost+found` directory. If the directory is successfully reconnected, a CONNECTED message is displayed. Otherwise, one of the `lost+found` error messages is displayed.

`n` – To ignore this error condition. This error causes the UNREF error condition in phase 4.

Phase 4: Check Reference Counts Messages

This phase checks the link count information obtained in phases 2 and 3. It reports error conditions resulting from:

- Unreferenced files
- A 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 and directories
- Incorrect total free-inode counts

All errors in this phase (except running out of space in the `lost+found` directory) are correctable when the file system is being preened.

```
BAD/DUP type I=inode-number OWNER=UID MODE=file-mode SIZE=file-size  
MTIME=modification-time (CLEAR)
```

Phase 1 or phase 1B found duplicate blocks or bad blocks associated with file or directory inode *inode-number*. The owner *UID*, mode *file-mode*, size *file-size*, and modify time *modification-time* of inode *inode-number* are displayed. This error does not display in this phase when the file system is being preened because `fsck` would have terminated earlier.

At the CLEAR prompt, type:

y - To deallocate inode *inode-number* by zeroing its contents.

n - To ignore this error condition.

(CLEAR)

The inode mentioned in the UNREF error message immediately preceding cannot be reconnected. This message does not display if the file system is being preened because lack of space to reconnect files terminates `fsck`.

At the CLEAR prompt, type:

y - To deallocate the inode by zeroing out its contents.

n - To ignore the preceding error condition.

```
LINK COUNT type I=inode-number OWNER=UID MODE=file-mode SIZE=file-size
MTIME=modification-time COUNT link-count SHOULD BE corrected-link-count
(ADJUST)
```

The link count for directory or file inode *inode-number* is *link-count* but should be *corrected-link-count*. The owner *UID*, mode *file-mode*, size *file-size*, and modify time *modification-time* of inode *inode-number* are displayed. If the `-o p` option is specified, the link count is adjusted unless the number of references is increasing. This condition does not occur unless there is a hardware failure. When the number of references is increasing during preening, `fsck` displays this message and exits:

```
LINK COUNT INCREASING
```

At the ADJUST prompt, type:

y - To replace the link count of directory or file inode *inode-number* with *corrected-link-count*.

n - To ignore this error condition.

```
lost+found IS NOT A DIRECTORY (REALLOCATE)
```

The entry for `lost+found` is not a directory.

At the `REALLOCATE` prompt, type:

`y` - To 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. It will either be reclaimed as an unreferenced inode or have its link count adjusted later in this phase. Inability to create a `lost+found` directory displays this message:

```
SORRY. CANNOT CREATE lost+found DIRECTORY
```

and aborts the attempt to link up the lost inode. This error generates the `UNREF` error message later in phase 4.

`n` - To abort the attempt to link up the lost inode. This error generates the `UNREF` error message later 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.

At the `CREATE` prompt, type:

`y` - To create a `lost+found` directory in the root of the file system. If the `lost+found` directory cannot be created, `fsck` displays the message:

```
SORRY. CANNOT CREATE lost+found DIRECTORY
```

and aborts the attempt to link up the lost inode. This error in turn generates the `UNREF` error message later in phase 4.

`n` - To abort the attempt to link up the lost inode. This generates the `UNREF` error message later 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.

At the EXPAND prompt, type:

y - To expand the `lost+found` directory to make room for the new entry. If the attempted expansion fails, `fsck` displays the message:

```
SORRY. NO SPACE IN lost+found DIRECTORY
```

and aborts the request to link a file to the `lost+found` directory. This error generates the UNREF error message later in phase 4. Delete any unnecessary entries in the `lost+found` directory. This error terminates `fsck` when preening (`-o p` option) is in effect.

n - To abort the attempt to link up the lost inode. This results in the UNREF error message later in phase 4.

```
UNREF FILE I=inode-number OWNER=UID MODE=file-mode SIZE=file-size  
MTIME=modification-time (RECONNECT)
```

File inode *inode-number*, was not connected to a directory entry when the file system was traversed. The owner *UID*, mode *file-mode*, size *file-size*, and modify time *modification-time* of inode *inode-number* are displayed. When `fsck` is preening, the file is cleared if either its size or its link count is zero; otherwise, it is reconnected.

At the RECONNECT prompt, type:

y - To reconnect inode *inode-number* to the file system in the `lost+found` directory. This error may generate the `lost+found` error message in phase 4 if there are problems connecting inode *inode-number* to the `lost+found` directory.

n - To ignore this error condition. This error always invokes the CLEAR error condition in phase 4.

```
UNREF type I=inode-number OWNER=UID MODE=file-mode SIZE=file-size  
MTIME=modification-time (CLEAR)
```

Inode *inode-number* (whose *type* is directory or file) was not connected to a directory entry when the file system was traversed. The owner *UID*, mode *file-mode*, size *file-size*, and modify time *modification-time* of inode *inode-number* are displayed. When `fsck` is preening, the file is cleared if either its size or its link count is zero; otherwise, it is reconnected.

At the CLEAR prompt, type:

y - To deallocate inode *inode-number* by zeroing its contents.

n - To ignore this error condition.

```
ZERO LENGTH DIRECTORY I=inode-number OWNER=UID MODE=file-mode  
SIZE=file-size MTIME=modification-time (CLEAR)
```

A directory entry *filename* has a size *file-size* that is zero. The owner *UID*, mode *file-mode*, size *file-size*, modify time *modification-time*, and directory name *filename* are displayed.

At the CLEAR prompt, type:

y - To deallocate the directory inode *inode-number* by zeroing out its contents.

n - To ignore the error condition.

Phase 5: Check Cylinder Groups Messages

This phase checks the free-block and used-inode maps. It reports error conditions resulting from:

- Allocated inodes missing from used-inode maps
- Free blocks missing from free-block maps
- Free inodes in the used-inode maps
- Incorrect total free-block count
- Incorrect total used inode count

```
BLK(S) MISSING IN BIT MAPS (SALVAGE)
```

A cylinder group block map is missing some free blocks. During preening, *fsck* reconstructs the maps.

At the SALVAGE prompt, type:

y - To reconstruct the free-block map.

n - To ignore this error condition.

CG *character-for-command-option*: BAD MAGIC NUMBER

The magic number of cylinder group *character-for-command-option* is wrong. This error usually indicates that the cylinder group maps have been destroyed. When running interactively, the cylinder group is marked as needing reconstruction. `fsck` terminates if the file system is being preened.

FREE BLK COUNT(S) WRONG IN SUPERBLK (SALVAGE)

The actual count of free blocks does not match the count of free blocks in the superblock of the file system. If the `-o p` option was specified, the free-block count in the superblock is fixed automatically.

At the SALVAGE prompt, type:

y - To reconstruct the superblock free-block information.

n - To ignore this error condition.

SUMMARY INFORMATION BAD (SALVAGE)

The summary information is incorrect. When preening, `fsck` recomputes the summary information.

At the SALVAGE prompt, type:

y - To reconstruct the summary information.

n - To ignore this error condition.

Cleanup Phase Messages

Once a file system has been checked, a few cleanup functions are performed. The cleanup phase displays the following status messages.

number-of files, *number-of-files* used, *number-of-files* free (*number-of* frags, *number-of* blocks, *percent* fragmentation)

This message indicates that the file system checked contains *number-of* files using *number-of* fragment-sized blocks, and that there are *number-of* fragment-sized blocks free in the file system. The numbers in parentheses break the free count down into *number-of* free fragments, *number-of* free full-sized blocks, and the *percent* fragmentation.

```
***** FILE SYSTEM WAS MODIFIED *****
```

This message indicates that the file system was modified by `fsck`. If this file system is mounted or is the current root file system, 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.

```
filename FILE SYSTEM STATE SET TO OKAY
```

This message indicates that file system *filename* was marked as stable. `fsck` with the `-m` option uses this information to determine that the file system does not need checking.

```
filename FILE SYSTEM STATE NOT SET TO OKAY
```

This message indicates that file system *filename* was *not* marked as stable. `fsck` with the `-m` option uses this information to determine that the file system needs checking.

Modifying Automatic Boot Checking

During boot up, a preliminary check on each file system to be mounted from a hard disk is run using the boot script `/sbin/rcS`, which checks the `/`, `/usr`, `/usr/kvm` file systems. The other `rc` shell scripts then use the `fsck` command to check each additional file system sequentially. They do not check file systems in parallel. File systems are checked sequentially during booting even if the `fsck pass` numbers are greater than one.

The `/etc/vfstab` File

When the commands for checking and mounting file systems are run without specifying a file system directly, the commands step through the file system table (`/etc/vfstab`) using the information specified in the various fields. The `fsck pass` field specifies information for file system checking. The `automount` field specifies information for mounting the file system at boot time.

When you create new file systems, add entries to `/etc/vfstab` indicating whether they are to be checked and mounted at boot time. See Chapter 3, “Mounting and Unmounting File Systems,” for more information about adding entries to the `/etc/vfstab` file.

Information in the `/etc/vfstab` file is specific for the slices and file systems for each system. Here is an example of an `/etc/vfstab` file:

```

venus% more /etc/vfstab
#device          device          mount          FS          fsck          auto-          mount
#to mount        to fsck         point          type         pass          mount?        options#
#/dev/dsk/c1d0s2 /dev/rdisk/c1d0s2 /usr           ufs          1             yes           -
/proc            -               /proc          proc         -             no            -
fd               -               /dev/fd        fd           -             no            -
swap             -               /tmp           tmpfs        -             yes           -

/dev/dsk/c0t0d0s0 /dev/rdisk/c0t0d0s0 /           ufs          1             no            -
/dev/dsk/c0t0d0s1 -               -              swap         -             no            -
/dev/dsk/c0t0d0s6 /dev/rdisk/c0t0d0s6 /usr          ufs          2             no            -
/dev/dsk/c0t0d0s7 /dev/rdisk/c0t0d0s7 /opt          ufs          3             yes           -
swsvr4-50:/export/svr4/man - /usr/man     nfs          no            yes           -
venus%

```

If you put a hyphen (-) in the `fsck pass` (the pass number) field, the generic `fsck` command will not check the file system regardless of the state of the file system. Use a hyphen in the `fsck pass` field for read-only file systems, remote file systems, or pseudo file systems, such as `/proc`, to which checking does not apply.

If you put a number (0 or greater) in the `fsck pass` field, the file system-specific `fsck` command is called. For UFS file systems, when the value is 0, the file system is not checked.

`fsck` automatically checks UFS file systems in parallel when the `fsck pass` value is greater than 1 and the `preen` option (`fsck -o p`) is used. The values can be any number greater than 1.

In `preen` mode, `fsck` allows only one active file system check per disk, starting a new check only after the previous one is completed. `fsck` automatically uses the major and minor numbers of the devices on which the file systems reside to determine how to check file systems on different disks at the same time.

When the `fsck pass` number is 1, file systems are checked sequentially, in the order they appear in the `/etc/vfstab` file. Usually, the root file system has the `fsck pass` set to 1.

Note – `fsck` does *not* use the `fsck pass` number to determine the sequence of file system checking.

▼ **How to Modify Automatic Checking Done During Booting**

1. **Become superuser.**
2. **Edit `/etc/vfstab` entries in the `fsck pass` field, and save the changes.**
The next time the system is booted, the new values are used.

Interactively Checking and Repairing a UFS File System

You may need to interactively check file systems:

- When they cannot be mounted
- When they develop problems while in use

When an in-use file system develops inconsistencies, error messages may be displayed in the console window or the system may crash.

Before using `fsck`, you may want to refer to “Syntax and Options for the `fsck` Command” on page 260 and “Error Messages” on page 226 for more information.

▼ **How to See if a File System Needs Checking**

1. **Become superuser.**
2. **Type `fsck -m /dev/rdisk/cntndnsn` and press Return.**
The state flag in the superblock of the file system you specify is checked to see whether the file system is clean or requires checking.

If you omit the device argument, all the UFS file systems listed in `/etc/vfstab` with a `fsck pass` value greater than 0 are checked.

In this example, the first file system needs checking; the second file system does not:

```
# fsck -m /dev/rdisk/c0t0d0s6
** /dev/rdisk/c0t0d0s6
ufs fsck: sanity check: /dev/rdisk/c0t0d0s6 needs checking
# fsck -m /dev/rdisk/c0t0d0s7
** /dev/rdisk/c0t0d0s7
ufs fsck: sanity check: /dev/rdisk/c0t0d0s7 okay
#
```

▼ How to Check File Systems Interactively

1. **Become superuser.**
2. **Unmount the file system.**
3. **Type `fsck` and press Return.**

All file systems in the `/etc/vfstab` file with entries in the `fsck` pass field greater than zero are checked. You can also specify the mount point directory or `/dev/rdisk/cntndnsn` as arguments to `fsck`. Any inconsistency messages are displayed. See “Error Messages” on page 226 for information about how to respond to the error message prompts to interactively check one or more UFS file systems:

In this example, `/dev/rdisk/c0t0d0s6` is checked and the incorrect block count is corrected:

```
# fsck /dev/rdisk/c0t0d0s6
checkfilesystem: /dev/rdisk/c0t0d0s6
** Phase 1 - Check Block and Sizes
INCORRECT BLOCK COUNT I=2529 (6 should be 2)
CORRECT? y

** Phase 2 - Check Pathnames
** Phase 3 - Check Connectivity
** Phase 4 - Check Reference Counts
** Phase 5 - Cylinder Groups
929 files, 8928 used, 2851 free (75 frags, 347 blocks, 0.6%
fragmentation)
```

```
/dev/rdsk/c0t0d0s6 FILE SYSTEM STATE SET TO OKAY  
  
***** FILE SYSTEM WAS MODIFIED *****
```

4. If you corrected any errors, type `fsck` and press Return.

`fsck` may not be able to fix all errors in one execution. If you see the message `FILE SYSTEM STATE NOT SET TO OKAY`, run the command again. If that does not work, see “How to Fix a UFS File System `fsck` Cannot Repair” on page 260.

5. Rename and move any files put in `lost+found`.

Individual files put in the `lost+found` directory by `fsck` are renamed with their inode numbers. If possible, rename the files and move them where they belong. You may be able to use the `grep` command to match phrases with individual files and the `file` command to identify file types. When whole directories are dumped into `lost+found`, it is easier to figure out where they belong and move them back.

Preening UFS File Systems

The `preen` option to `fsck` (`fsck -o p`) checks UFS file systems and automatically fixes only the simple problems that normally result from an unexpected system halt. It exits immediately if it encounters a problem that requires operator intervention. The `preen` option also permits parallel checking of file systems.

You can run `fsck` with the `-o p` option to preen the file systems after an unclean halt. In this mode, `fsck` does not look at the clean flag and does a full check. While preening a file system, `fsck` fixes only corruptions that are expected to result from an unclean halt. These actions are a subset of the actions that `fsck` takes when it runs interactively.

▼ How to Preen UFS File Systems

1. **Become superuser.**

2. **Unmount the UFS file systems.**

3. **Type `fsck -o p` and press Return.**

All file systems in the `/etc/vfstab` file with entries greater than zero in the `fsck pass` field are checked. Files with an `fsck pass` number greater than 1 are checked in parallel according to their major and minor numbers. Normal inconsistencies that occur when the CPU halts without warning are repaired. If `fsck` encounters a more serious error, it exits immediately.

You can preen individual file systems by using *mount-point* or `/dev/rdisk/cntndnsn` as arguments to `fsck`. For example, to preen the `/usr` file system, type `fsck -o p /usr` and press Return.

Restoring a Bad Superblock

When the superblock of a file system becomes damaged, you must restore it. `fsck` tells you when a superblock is bad. Fortunately, redundant copies of the superblock are stored within a file system. You can use `fsck -o b` to replace the superblock with one of the copies.

▼ How to Restore a Bad Superblock

1. **Become superuser.**

2. **Change to a directory outside the damaged file system.**

3. **Type `umount mount-point` and press Return.**



Caution – Be sure to use the `-N` option with `newfs` in the next step. If you omit the `-N` option, you will create a new, empty file system.

4. **Type `newfs -N /dev/rdisk/cntndnsn` and press Return.**

The output of this command displays the block numbers that were used for the superblock copies when `newfs` created the file system.

5. Type `fsck -F ufs -o b=block-number /dev/rdisk/cntndnsn` and press Return.

`fsck` uses the alternative superblock you specify to restore the primary superblock. You can always try 32 as an alternative block, or use any of the alternative blocks show by `newfs -N`.

In this example, superblock copy 5264 is restored for the `/files7` file system:

```
# cd /
# umount /files7
# newfs -N /dev/rdisk/c0t3d0s7
/dev/rdisk/c0t3d0s7: 163944 sectors in 506 cylinders of 9 tracks,
36 sectors
 83.9MB in 32 cyl groups (16 c/g, 2.65MB/g, 1216 i/g)
super-block backups (for fsck -b #) at:
 32, 5264, 10496, 15728, 20960, 26192, 31424, 36656, 41888,
 47120, 52352, 57584, 62816, 68048, 73280, 78512, 82976, 88208,
 93440, 98672, 103904, 109136, 114368, 119600, 124832, 130064,
135296,
140528, 145760, 150992, 156224, 161456,
# fsck -F ufs -o b=5264 /dev/rdisk/c0t3d0s7
Alternate superblock location: 5264.
** /dev/rdisk/c0t3d0s7
** Last Mounted on
** Phase 1 - Check Blocks and Sizes
** Phase 2 - Check Pathnames
** Phase 3 - Check Connectivity
** Phase 4 - Check Reference Counts
** Phase 5 - Check Cyl groups
36 files, 867 used, 75712 free (16 frags, 9462 blocks, 0.0%
fragmentation)
/dev/rdisk/c0t3d0s7 FILE SYSTEM STATE SET TO OKAY

***** FILE SYSTEM WAS MODIFIED *****
#
```

If the superblock in the root file system becomes damaged and you cannot boot the system, reinstall `/kernel/unix` and rebuild the root file system with `newfs`. Because a superblock is created by the `newfs` command, you do not need to restore it. See “Restoring Complete File Systems” on page 142 for information on how to restore `/` and `/usr` file systems.

How to Fix a UFS File System `fsck` Cannot Repair

Sometimes you need to run `fsck` a few times to fix a file system, because problems corrected on one pass may uncover other problems not found in earlier passes. `fsck` does not keep running until it comes up clean, so you must rerun it manually.

Pay attention to the information displayed by `fsck`. It may help you fix the problem. For example, the messages may point to a bad directory. If you delete the directory, you may find that `fsck` runs cleanly.

If `fsck` still cannot repair the file system, you can try to use the `fsdb`, `ff`, `clri`, and `ncheck` commands to figure out and fix what is wrong. See the `fsdb(1M)`, `ff(1M)`, `clri(1M)`, and `ncheck(1M)` manual pages for information about how to use these commands. You may, ultimately, need to re-create the file system and restore its contents from backup media. See Chapter 8, “Restoring Files and File Systems,” for information about restoring complete file systems.

If you cannot fully repair a file system but you can mount it read-only, try using `cp`, `tar`, or `cpio` to retrieve all or part of the data from the file system.

If hardware disk errors are causing the problem, you may need to reformat and divide the disk into slices again before re-creating and restoring file systems. Hardware errors usually display the same error again and again across different commands. The `format(1M)` command tries to work around bad blocks on the disk. If the disk is too severely damaged, however, the problems may persist, even after reformatting. See the `format(1M)` manual page for information about using the `format` command. See *Peripherals Administration* for information about installing a new disk.

Syntax and Options for the `fsck` Command

The `fsck` command checks and repairs inconsistencies in file systems. It has four options:

- Checks only whether a file system can be mounted (`fsck -m`)
- Interactively asks for confirmation before making repairs (`fsck`)

- Assumes yes or no response for all repairs (`fsck -y`)
- Noninteractively preens the file system, fixing all expected (innocuous) inconsistencies, but exiting when a serious problem is encountered (`fsck -o p`)

Generic `fsck` Command Syntax, Options, and Arguments

The `fsck` command has two components: a generic component and a component specific to each type of file system. The generic commands apply to most types of file systems, while the specific commands apply to only one type of file system. You should always use the generic command, which calls the file system-specific command, as needed.

Usually, you must be superuser to run `fsck`. You can run the `fsck` command without being superuser; but to make repairs, the file system should be unmounted and you must have read permission for the raw device file for the slice (a potential security hole).

The generic `fsck` command goes through `/etc/vfstab` to see what file systems to check. It runs the appropriate file system-specific `fsck` command on each file system listed, except those excluded by an `fsck` pass number of `-` or `0` (UFS only).

The generic `fsck` command has the following syntax:

```
/usr/sbin/fsck [-F type] [-V] [-m] [special]  
  
/usr/sbin/fsck [-F type] [-V] [-y|Y]|[n|N] [-o specific-options][special]
```

The generic `fsck` command has the following options and arguments:

`-F`

Specifies the file system type (*type*). If *type* is not specified on the command line, it is obtained from `/etc/vfstab` by matching an entry in that file with the *special* device name specified. If no entry is found, the default local file system type specified in `/etc/default/fs` is used.

-v

Echoes the completed command line (verbose). The echoed line includes additional information derived from `/etc/vfstab`. This option can be used to verify and validate the command line. It does not execute the command.

-m

Performs a preliminary (“sanity”) check only. It returns a code indicating the state of the file system: 0 for “clean” and 32 for “dirty.” This option is used by the startup script `/etc/bcheckrc` to determine whether a file system needs to be checked.

-y or -Y or -n or -N

Runs the command automatically answering yes or no to all prompts.

specific-options

A comma separated list of options that follow the `-o` option. This list describes the options that are passed to the UFS-specific `fsck` command for interpretation.

p

Preens. Runs the command automatically in silent mode, correcting what it can, but exiting when it encounters a problem that requires intervention. This option also enables parallel checking of UFS file systems.

b=*blocknumber*

Uses the alternative (redundant) superblock, located at the specified location. This option can be used to repair a bad superblock. You can display a list of alternative superblocks using the `newfs -N` command.

c

Converts an old format file system with statically allocated tables to new format dynamically allocated tables. Static allocation imposes a hard maximum on table size, while dynamic allocation means space for tables can be added as needed after the initial allocation. If the file system is in the new format, convert it to the old format, unless the table allocation exceeds the fixed maximum allowed in the old format. `fsck` lists the direction of the conversion. In interactive mode, `fsck` prompts for confirmation before doing the conversion. When you use the `-o p` option, the conversion is attempted without asking for confirmation. This option is useful when you want to convert a number of file systems at

once. You can determine whether a file system is in the old or new format by running the `fstyp(1M)` command, and looking at the first line displayed.

^w

Checks only file systems that permit write access.

special

Specifies the mount point or raw device name of one or more file systems. An entry for the mount point must exist in `/etc/vfstab`. If you omit the *special* argument, entries in `/etc/vfstab` with a specified `fsck` device and a `fsck pass` number greater than zero are checked. If preening (`-o p`) is in effect and more than one entry has an `fsck pass` number greater than 1, file systems on different disks are checked in parallel.

File System Reference



This appendix has these sections:

<i>Default Directories for / and /usr File Systems</i>	<i>page 265</i>
<i>The Structure of UFS File System Cylinder Groups</i>	<i>page 270</i>
<i>Deciding on Custom File System Parameters</i>	<i>page 274</i>
<i>Commands for Creating a Customized File System</i>	<i>page 278</i>

Default Directories for / and /usr File Systems

Table A-1 describes all the directories contained in the default root and /usr file systems. See the *Administration Supplement for Solaris Platforms* for a list of directories that are platform specific. See “The Default SunOS File Systems” on page 12 for a description of all of the directories in the default SunOS file system.

Table A-1 Default Directories for root and /usr File Systems (1 of 6)

Directory	Description
Directories in the root file system:	
/	Root of the overall file system name space
/dev	Primary location for special files
/dev/dsk	Block disk devices
/dev/pts	pty slave devices

Table A-1 Default Directories for `root` and `/usr` File Systems (2 of 6)

Directory	Description
<code>/dev/rdisk</code>	Raw disk devices
<code>/dev/rmt</code>	Raw tape devices
<code>/dev/sad</code>	Entry points for the STREAMS Administrative Driver
<code>/dev/term</code>	Terminal devices
<code>/etc</code>	Host-specific system administrative configuration files and databases
<code>/etc/acct</code>	Accounting system configuration information
<code>/etc/cron.d</code>	Configuration information for <code>cron</code>
<code>/etc/default</code>	Defaults information for various programs
<code>/etc/dfs</code>	Configuration information for exported file systems
<code>/etc/fs</code>	Binaries organized by fs types for operations required before <code>/usr</code> is mounted.
<code>/etc/inet</code>	Configuration files for Internet services
<code>/etc/init.d</code>	Scripts for changing between run levels
<code>/etc/lp</code>	Configuration information for the printer subsystem
<code>/etc/mail</code>	Mail subsystem configuration
<code>/etc/net</code>	Configuration information for <code>ti</code> (transport-independent) network services
<code>/etc/opt</code>	Configuration information for optional packages
<code>/etc/rc0.d</code>	Scripts for entering/leaving run level 0
<code>/etc/rc1.d</code>	Scripts for entering/leaving run level 1
<code>/etc/rc2.d</code>	Scripts for entering/leaving run level 2
<code>/etc/rc3.d</code>	Scripts for entering/leaving run level 3
<code>/etc/rcS.d</code>	Scripts for bringing the system up in single user mode
<code>/etc/saf</code>	Service access facility files (including FIFOs)
<code>/etc/skel</code>	Default profile scripts for new user accounts
<code>/etc/sm</code>	Status monitor information
<code>/etc/sm.bak</code>	Backup copy of status monitor information

Table A-1 Default Directories for `root` and `/usr` File Systems (3 of 6)

Directory	Description
<code>/etc/tm</code>	Trademark files; contents displayed at boot time
<code>/etc/uucp</code>	uucp configuration information
<code>/export</code>	Default root of the exported file system tree
<code>/home</code>	Default root of a subtree for user directories
<code>/kernel</code>	Subtree of loadable kernel modules, including the base kernel itself as <code>/kernel/unix</code>
<code>/mnt</code>	Convenient, temporary mount point for file systems
<code>/opt</code>	Root of a subtree for add-on application packages
<code>/opt/SUNWspro</code>	Mount/installation point for unbundled language products
<code>/sbin</code>	Essential executables used in the booting process and in manual system failure recovery
<code>/stand</code>	Standalone programs
<code>/tmp</code>	Temporary files; cleared during boot sequence
<code>/usr</code>	Mount point for <code>/usr</code> file system
<code>/var</code>	Root of a subtree of varying files
<code>/var/adm</code>	System logging and accounting files
<code>/var/crash</code>	Default depository for kernel crash dumps
<code>/var/cron</code>	<code>cron</code> 's log file
<code>/var/lp</code>	Line printer subsystem logging information
<code>/var/mail</code>	Directory where users' mail is kept
<code>/var/news</code>	Community service messages (<i>note</i> : not the same as USENET-style news)
<code>/var/nis</code>	NIS+ databases
<code>/var/opt</code>	Root of a subtree for varying files associated with software packages
<code>/var/preserve</code>	Backup files for <code>vi</code> and <code>ex</code>
<code>/var/sadm</code>	Databases maintained by the software package management utilities

Table A-1 Default Directories for root and /usr File Systems (4 of 6)

Directory	Description
/var/saf	saf (service access facility) logging and accounting files
/var/spool	Directories for spooled temporary files
/var/spool/cron	cron and at spool files
/var/spool/locks	Spooling lock files
/var/spool/lp	Line printer spool files
/var/spool/mqueue	Mail queued for delivery
/var/spool/pkg	Spooled packages
/var/spool/uucp	Queued uucp jobs
/var/spool/uucppublic	Files deposited by uucp
/var/tmp	Directory for temporary files; not cleared during boot sequence
/var/uucp	c log and status files
/var/yp	NIS databases (for backwards compatibility with NIS and unnecessary after full transition to NIS+)

Directories in the /usr file system

bin	Location for standard system commands
demo	Demo programs and data
games	Game binaries and data
include	Header files (for C programs, etc.)
kernel	Additional modules
kvm	Implementation architecture-specific binaries and libraries
lib	Various program libraries, architecture-dependent databases, and binaries not invoked directly by the user
lib/acct	Accounting scripts and binaries
lib/class	Scheduling class-specific directories containing executables for priocntl and dispadmin commands

Table A-1 Default Directories for `root` and `/usr` File Systems (5 of 6)

Directory	Description
<code>lib/font</code>	<code>troff</code> font description files
<code>lib/fs</code>	File system type-dependent modules; not invoked directly by the user
<code>lib/iconv</code>	Conversion tables for <code>iconv(1)</code>
<code>lib/libp</code>	Profiled libraries
<code>lib/locale</code>	Internationalization localization databases
<code>lib/lp</code>	Line printer subsystem databases and back-end executables
<code>lib/mail</code>	Auxiliary programs for the <code>mail</code> subsystem
<code>lib/netsvc</code>	Internet network services
<code>lib/nfs</code>	Auxiliary NFS-related programs and daemons
<code>lib/pics</code>	PIC archives needed to build the run-time linker
<code>lib/refer</code>	Auxiliary <code>refer</code> -related programs
<code>lib/sa</code>	Scripts and commands for the system activity report package
<code>lib/saf</code>	Auxiliary programs and daemons related to the service access facility
<code>lib/uucp</code>	Auxiliary <code>uucp</code> -related programs and daemons
<code>lib/zoneinfo</code>	Time zone information
<code>local</code>	Commands local to a site
<code>old</code>	Programs that are being phased out
<code>openwin</code>	Mount/installation point for OpenWindows software
<code>sadm</code>	Various files and directories related to system administration; see specifics below
<code>sadm/bin</code>	"valtools" binaries for use by FMLI scripts
<code>sadm/install</code>	Executables and scripts for <code>pkg</code> management
<code>sbin</code>	Executables for system administration
<code>sbin/static</code>	Statically linked version of selected programs from <code>/usr/bin</code> and <code>/usr/sbin</code>

Table A-1 Default Directories for root and /usr File Systems (6 of 6)

Directory	Description
share	Architecture-independent sharable files
share/lib	Architecture-independent databases
share/lib/keytables	Keyboard layout description tables
share/lib/mailx	mailx-related help files
share/lib/nterm	nroff terminal tables
share/lib/pub	Various data files
share/lib/spell	Auxiliary spell-related databases and scripts
share/lib/tabset	Tab setting escape sequences
share/lib/terminfo	terminfo-style terminal description files
share/lib/tmac	[nt]roff macro packages
share/src	Source code for kernel, libraries, and utilities
ucb	Berkeley compatibility package binaries
ucbinclude	Berkeley compatibility package header files
ucblib	Berkeley compatibility package libraries

The Structure of UFS File System Cylinder Groups

When you create a UFS file system, the disk slice is divided into cylinder groups, which are then divided into blocks to control and organize the structure of the files within the *cylinder group*. A cylinder group has one or more consecutive disk cylinders. Each type of block has a specific function in the file system. A UFS file system has these four types of blocks:

- Boot block – Used to store information used when booting the system
- Superblock – Used to store much of the information about the file system
- Inode – Used to store all information about a file except its name
- Storage or data block – Used to store data for each file

This section provides additional information about the organization and function of these blocks.

The Boot Block

The boot block stores 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 boot block appears only in the first cylinder group (cylinder group 0) and is the first 8 Kbytes in a partition.

The Superblock

The superblock stores much of the information about the file system. A few of the more important things it contains are:

- Size and status of the file system
- Label (file system name and volume name)
- Size of the file system logical block
- Date and time of the last update
- Cylinder group size
- Number of data blocks in a cylinder group
- Summary data block
- File system state: clean, stable, or active
- Path name of the last mount point

The superblock is located at the beginning of the disk partition, and is replicated in each cylinder group. Because the superblock contains critical data, multiple superblocks are made when the file system is created. Each of the superblock replicas is offset by a different amount from the beginning of its cylinder group. For multiple-platter disk drives, the offsets are calculated so that a superblock appears on each platter of the drive. That way, if the first platter is lost, an alternate superblock can always be retrieved. Except for the leading blocks in the first cylinder group, the leading blocks created by the offsets are used for data storage.

A summary information block is kept with the superblock. It is not replicated, but is grouped with the first superblock, usually in cylinder group 0. The summary block records changes that take place as the file system is used, and lists the number of inodes, directories, fragments, and storage blocks within the file system.

Inodes

An inode contains all the information about a file except its name, which is kept in a directory. An inode is 128 bytes. The inode information is kept in the cylinder information block, and contains:

- The type of the file
 - Regular
 - Directory
 - Block special
 - Character special
 - Symbolic link
 - FIFO, also known as named pipe
 - Socket
- The mode of the file (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 (0 to 14) point to the data blocks that store the contents of the file. 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 block addresses instead of file contents. The 14th address points to a double indirect block, which contains addresses of indirect blocks. The 15th address is for triple indirect addresses, if they are ever needed. Figure A-1 shows this chaining of address blocks starting from the inode.

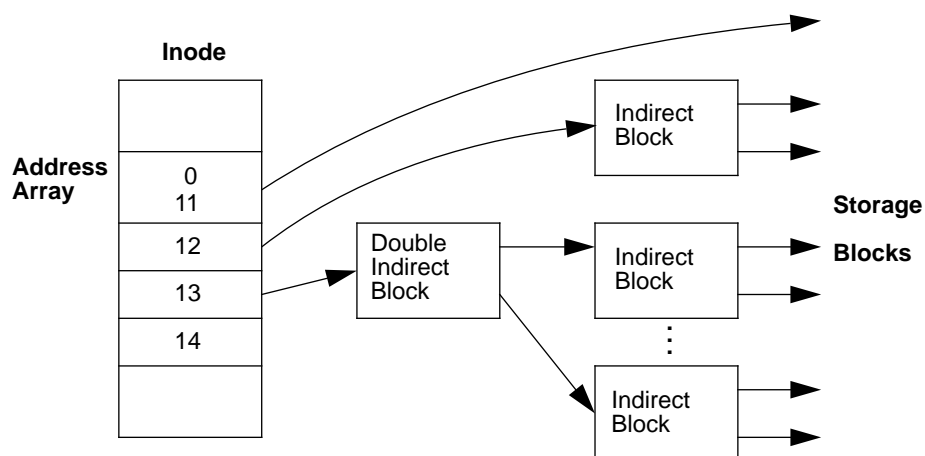


Figure A-1 The File System Address Chain in a UFS System

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. Storage blocks are allocated, by default, in two sizes: an 8-Kbyte logical block size, and a 1-Kbyte fragmentation size.

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 file name of the files in the directory.

Free Blocks

Blocks not currently being used as inodes, as indirect address blocks, or as storage blocks are marked as free in the cylinder group map. This map also keeps track of fragments to prevent fragmentation from degrading disk performance.

To give you an idea of the appearance of a typical UFS file system, Figure A-2 shows a series of cylinder groups in a generic UFS file system.

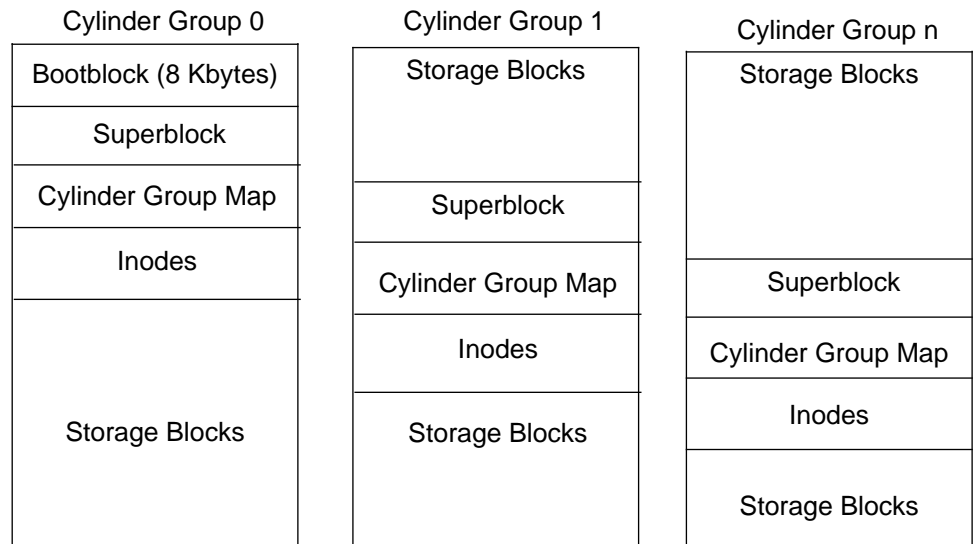


Figure A-2 A Typical UFS File System

Deciding on Custom File System Parameters

If you choose to alter the default file system parameters assigned by the `newfs` command, you need to understand them so you can decide whether to use the default values or to change them. This section describes each of these parameters:

- Block size
- Fragment size
- Minimum free space
- Rotational delay
- Optimization type
- Number of inodes

Logical Block Size

The logical block size is the size of the blocks that the UNIX kernel uses to read or write files. The logical block size is usually different from the physical block size (usually 512 bytes), which is the size of the smallest block that the disk controller can read or write.

You can specify the logical block size of the file system. Once the file system is created, you cannot change this parameter without rebuilding the file system. You can have file systems with different logical block sizes on the same disk.

By default, the logical block size is 8192 bytes (8 Kbytes) for UFS file systems. The UFS file system supports block sizes of 4096 or 8192 bytes (4 or 8 Kbytes). 8 Kbytes is the recommended logical block size.

To choose the best logical block size for your system, consider both the performance desired and the available space. For most UFS systems, an 8 Kbyte file system provides the best performance, offering a good balance between disk performance and use of space in primary memory and on disk.

As a general rule, to increase efficiency, use a larger logical block size for file systems where most of the files are very large. Use a smaller logical block size for file systems where most of the files are very small. You can use the `quot -c file-system` command on a file system to display a complete report on the distribution of files by block size.

Fragment Size

As files are created or expanded, they are allocated disk space in either full logical blocks or portions of logical blocks called fragments. When disk space is needed to hold a data for a file, full blocks are allocated first, and then one or more fragments of a block are allocated for the remainder. For small files allocation begins with fragments.

The ability to allocate fragments of blocks to files, rather than just whole blocks saves space by reducing *fragmentation* of disk space resulting from unused holes in blocks.

You define the *fragment size* when you create a UFS file system. The default fragment size is 1 Kbyte. Each block can be divided into 1, 2, 4, or 8 fragments, which results in fragment sizes from 8192 bytes to 512 bytes (for 4-Kbyte file systems only). The lower bound is actually tied to the disk sector size, typically 512 bytes.

Note – The upper bound may equal the full block size, in which case the fragment is not a fragment at all. This configuration may be optimal for file systems with very large files when you are more concerned with speed than with space.

When choosing a fragment size, look at the trade-off between time and space: a small fragment size saves space, but requires more time to allocate. As a general rule, to increase storage efficiency, use a larger fragment size for file systems where most of the files are large. Use a smaller fragment size for file systems where most of the files are small.

Minimum Free Space

The *minimum free space* is the percentage of the total disk space held in reserve when you create the file system. The default reserve is 10 percent. Free space is important because file access becomes less and less efficient as a file system gets full. As long as there is an adequate amount of free space, UFS file systems operate efficiently. When a file system becomes full, using up the available user space, only the superuser can access the reserved free space.

Commands such as `df` report the percentage of space that is available to users, excluding the percentage allocated as the minimum free space. When the command reports that more than 100 percent of the disk space in the file system is in use, some of the reserve has been used by root.

If you impose quotas on users, the amount of space available to the users does not include the free space reserve. You can change the value of the minimum free space for an existing file system using the `tunefs` command.

Rotational Delay (Gap)

The *rotational delay* is the expected minimum time (in milliseconds) it takes the CPU to complete a data transfer and initiate a new data transfer on the same disk cylinder. The default delay depends on the type of the disk, and is usually optimized for each disk type.

When writing a file, the UFS allocation routines try to position new blocks on the same disk cylinder as the previous block in the same file. The allocation routines also try to optimally position new blocks within tracks to minimize the disk rotation needed to access them.

To position file blocks so they are “rotationally well-behaved,” the allocation routines must know how fast the CPU can service transfers and how long it takes the disk to skip over a block. Using options to the `mkfs` command, you can indicate how fast the disk rotates and how many disk blocks (sectors) it has per track. The allocation routines use this information to figure out how many milliseconds it takes to skip a disk block. Then using the expected transfer time (rotational delay), blocks can be optimally positioned or spaced so that the next block is just coming under the disk head when the system is ready to read it.

Note – It is not necessary to specify the rotational delay (`-d` option to `newfs`) for some devices.

Place blocks consecutively only if your system is fast enough to read them on the same disk rotation. If the system is too slow, the disk spins past the beginning of the next block in the file and must complete a full rotation before the block can be read, which takes a lot of time. You should try to specify an appropriate value for the gap so that the head is located over the appropriate block when the next disk request occurs.

You can change the value of this parameter for an existing file system using the `tunefs` command. The change applies only to subsequent block allocation, not to blocks already allocated.

Optimization Type

The *optimization type* is either *space* or *time*.

- *Space* – When you select *space* optimization, disk blocks are allocated to minimize fragmentation and disk use is optimized. Space is the default when you set the minimum free space to less than 10 percent.
- *Time* – When you select *time* optimization, disk blocks are allocated as quickly as possible, with less emphasis on their placement. Time is the default when you set the minimum free space to 10 percent or greater. When there is enough free space, it is relatively easy to allocate disk blocks well, without resulting in too much fragmentation.

You can change the value of this parameter for an existing file system using the `tunefs` command.

Number of Bytes per Inode

The number of inodes determines the number of files you can have in the file system: one inode for each file. The *number of bytes per inode* determines the total number of inodes created when the file system is made: the total size of the file system divided by the number of bytes per inode. Once the inodes are allocated, you cannot change the number without re-creating the file system.

The default number of bytes per inode is 2048 bytes (2 Kbytes), which assumes the average size of each file is 2 Kbytes or greater. Most files are larger than 2 Kbytes. If you have a file system with many symbolic links, they can lower the average file size. If your file system is going to have many small files, you can give this parameter a lower value. Note, however, that having too many inodes is much better than running out of them. If you have too few inodes, you could reach the maximum number of files on a disk partition that is practically empty.

Commands for Creating a Customized File System

This section describes the two commands you use to create a customized file system:

- `newfs`
- `mkfs`

The `newfs` Command Syntax, Options, and Arguments

The `newfs` command is located in `/usr/sbin/newfs`.

- t *ntrack*
The number of tracks per cylinder on the disk. The default is determined from the disk label.
- b *bsize*
The logical block size in bytes to use for data transfers. Specify the size of 4096 or 8192 (4 or 8 Kbytes). The default is 8192 bytes (8 Kbytes).
- f *fragsize*
The smallest amount of disk space in bytes that is allocated to a file. Specify the fragment size in powers of two in the range from 512 to 8192 bytes. The default is 1024 bytes (1 Kbyte).
- c *cgsiz*e
The number of disk cylinders per cylinder group. This number must be in the range 1 to 32. The default is 16.
- m *free*
The minimum percentage of free disk space to allow. The default is 10 percent.
- r *rpm*
The speed of the disk, in revolutions per minute. The default is 3600. This parameter is converted to revolutions per second before it is passed to `mkfs`.
- i *nbpi*
The number of bytes per inode to use in computing how many inodes to create. The default is 2048.
- o *opt*
Optimization type to use for allocating disk blocks to files: `s` for space or `t` for time.
- a *apc*
The number of alternate blocks per disk cylinder (SCSI devices only) to reserve for bad block placement. The default is 0.
- d *gap*
(Rotational delay.) The expected minimum number of milliseconds it takes the CPU to complete a data transfer and initiate a new data transfer on the same disk cylinder. The default is 4.

-d *nrpos*

The number of different rotation positions in which to divide a cylinder group. The default is 8.

-C *maxcontig*

The maximum number of blocks, belonging to one file, that will be allocated contiguously before inserting a rotational delay. The default varies from drive to drive. Drives without internal (track) buffers (or drives/controllers that don't advertise the existence of an internal buffer) default to 1. Drives with buffers default to 7.

This parameter is limited in the following way:

*blocksize * maxcontig* must be \leq *maxphys*

maxphys is a read-only kernel variable that specifies the maximum block transfer size (in bytes) that the I/O subsystem is capable of satisfying. (This limit is enforced by `mount`, not by `newfs` or `mkfs`.)

This parameter also controls clustering. Regardless of the value of *rotdelay*, clustering is enabled only when *maxcontig* is greater than 1. Clustering allows higher I/O rates for sequential I/O and is described in the `tunefs(1M)` manual page.

The following argument is required:

special

The special character (raw) device file name of the partition to contain the file system.

The Generic `mkfs` Command

The generic `mkfs` command calls a file system-specific `mkfs`, which then creates a file system of a specified type on a specified disk partition. Although `mkfs` can support different types of file systems, in practice you would use it to create UFS file systems. To make other types of file systems, you would have to write the software for the file system-specific versions of the `mkfs` command to use. Normally, you do not run `mkfs` directly; it is called by the `newfs` command.

The generic `mkfs` command is located in `/usr/sbin`. See the `mkfs(1M)` manual page for a description of the arguments and options.

≡ A

Bibliography



This appendix contains a list of books that describe system administration and UNIX System V Release 4. Although these books are not specific to the SunOS release, you may find them to be useful.

General References

UNIX System V Release 4: The Complete Reference. Stephen Coffin. Osborne McGraw-Hill, 1990.

UNIX System V Release 4: An Introduction. Kenneth H. Rosen, Richard R. Rosinski, and James M. Farber. Osborne McGraw-Hill, 1990.

UNIX System Administration Handbook. Evi Nemeth, Garth Snyder, and Scott Seebass. Prentice Hall Software Series, 1989.

System Performance Tuning. Mike Loukides. O'Reilly & Associates, 1990.

Managing NFS and NIS. Hal Stern. O'Reilly & Associates, 1991.

Practical UNIX Security. Simson Garfinkel and Gene Spifford. O'Reilly & Associates, 1991.

Electronic Mail References

!%@:: A Directory of Electronic Mail Addressing & Network. 2d ed. Donnalyn Frey and Rick Adams. O'Reilly & Associates, 1990.

The DDN Protocol Handbook. Three-volume set of RFCs, available from SRI International, 333 Ravenswood Avenue, Menlo Park, CA 94025.

RFC 1211 *Problems with Maintaining Large email Lists.*

RFC 822 *Standard for the Format of ARPA INTERNET Text Messages.*

Shell References

The UNIX C Shell Field Guide. Gail Anderson and Paul Anderson. Prentice Hall, 1986.

Programming Languages

The AWK Programming Language. Alfred V. Aho, Brian W. Kernighan, and Peter J. Weinberger. Addison-Wesley, 1988.

sed & awk. Dale Dougherty. O'Reilly & Associates, 1991.

Virtual Swap Space Reference

Virtual Swap Space in SunOS, a white paper by Howard Chartock and Peter Snyder, Sun Microsystems, Inc.

Glossary

Administration Tool

A tool from which you can access various applications for administering systems.

archive

A copy of files on secondary media. The files are removed from the system because they are no longer active.

autofs

Software that automatically mounts a directory when a user changes into it, and unmounts the directory when it is no longer in use.

auto_home database

The database that you use to add home directories to `autofs`. You access the `auto_home` database using the Database Manager.

backup schedule

The schedule you establish for a site that determines when you run the `ufsdump` command. Use `ufsdump` regularly at different levels to back up user files and essential file systems.

bang

An exclamation point (!) that acts as a single-character UNIX command or as a separator between the routes of a route-based email address.

boot block

An 8-Kbyte disk block that contains information used during booting, including block numbers that point to the location of the `/boot` program on the disk. The boot block directly follows the disk label.

booting

The process of powering up a system, testing to determine which attached hardware devices are running, and bringing the operating system kernel into memory and operation at the run level specified by the boot command.

cache

A small, fast memory area that holds the most active part of a larger and slower memory.

core file

An image of the state of the software when it failed, used for troubleshooting. `core` files can be created by any software, including the operating system kernel.

crash

See **hang**.

cylinder group

One or more consecutive disk cylinders that include inode slots for files.

cylinder group map

A bit map in a UFS file system that stores information about block use and availability within each cylinder. The cylinder group replaces the traditional free list.

crash dump

A `core` file image of the operating system kernel that is saved in the swap partition when a system crashes. If crash dumps are enabled, the `core` image is written from the swap partition to a file.

daemon

A special type of program that, once activated, starts itself and carries out a specific task without any need for user input. Daemons are typically used to handle jobs that have been queued such as printing, mail, and communication.

Database Manager

A tool accessed from the Administration Tool software, which is used to administer NIS+ tables and UFS files in the `/etc` directory. You can also use the Database Manager to look at (but not edit) the contents of NIS maps.

diskette

A nonvolatile storage medium used to store and access data magnetically. SunOS system software supports 3.5-inch double-sided high density (DS, HD) diskettes.

disk quotas

A mechanism for controlling how much of a file system's resources any individual user can access. Disk quotas are optional and must be configured and administered to be used.

diskless client

A system with no local disk drive, that relies on an NFS server for the operating system, swap space, file storage, and other basic services.

domain

A directory structure for electronic mail addressing and network address naming. Within the United States, top-level domains include *com* for commercial organizations, *edu* for educational organizations, *gov* for governments, *mil* for the military, *net* for networking organizations, and *org* for other organizations. Outside of the United States, top-level domains designate the country. Subdomains designate the organization and the individual system.

domain addressing

Using a domain address to specify the destination of an electronic mail message.

DS, HD

Double-sided, high density. The type of 3.5-inch diskettes supported by the SunOS system software.

dump

The process of copying directories onto media (usually tape) for offline storage, using the `ufsdump` command.

electronic mail

A set of programs that transmit mail messages from one system to another, usually over communications lines. Electronic mail is frequently referred to as *email*.

email

See **electronic mail**.

environment variable	A system- or user-defined variable that provides information about the operating environment to the shell.
file system	A hierarchical arrangement of directories and files.
floppy diskette	See diskette .
free list	See cylinder group map .
full backup	A complete, level 0 backup of a file system done using the <code>ufsdump</code> command.
fully qualified domain name	A domain name that contains all of the elements needed to specify where an electronic mail message should be delivered. See also domain .
gateway	A system that handles electronic mail traffic between differing communications networks.
GID	The group identification number used by the system to control access to accounts owned by other users.
group database	The database that you use to create new group accounts or to modify existing group accounts. You access the group database from the Database Manager.
hang	A condition where a system does not respond to input from the keyboard or mouse.
hard limit	For disk quotas, a maximum limit on file system resources (blocks and inodes) that users cannot exceed.
home directory	The part of the file system allocated to an individual user for private files.

hosts database

The database that you use to control network security. You access the hosts database from the Database Manager.

incremental backup

A partial backup of a file system using the `ufsdump` command that includes only those files in the specified file system that have changed since a previous backup at a lower level.

initialization files

The “dot” files (files prefixed with “.”) in a user’s home directory that set the path, environment variables, windowing environment, and other characteristics to get users up and functioning.

init state

One of the seven initialization states, or run levels, a system can be running. A system can run in only one init state at a time.

inode

An entry in a predesignated area of a disk that describes where a file is located on that disk, the size of the file, when it was last used, and other identification information.

IP address

A unique internet protocol number that identifies each system in a network.

kernel

The master program set of SunOS software that manages all the physical resources of the computer, including file system management, virtual memory, reading and writing files to disks and tapes, scheduling of processes, printing, and communicating over a network.

login name

The name assigned to an individual user that controls access to a system.

monitor

The program in the PROM that provides a limited set of commands that can be used before the kernel is available.

mount point

A directory in the file system hierarchy where another file system is attached to the hierarchy.

NIS	The SunOS 4.x network information service.
NIS+	The SunOS 5.x network information service.
NFS	The default SunOS distributed file system that provides file sharing among systems. NFS servers can also provide kernels and swap files to diskless clients.
parse	To divide a string of characters or a series of words into parts to determine their collective meaning. Virtually every program that accepts command input must do some sort of parsing before the commands can be acted upon. For example, the <code>sendmail</code> program divides an email address into parts to decide where to send the message.
partition	A discrete portion of a disk, configured using the <code>format</code> program.
passwd database	The database that you use to add, modify, or delete user accounts. You access the passwd database from the Database Manager.
path	The list of directories that are searched to find an executable command.
path name	A list of directory names, separated with slashes (/), that specifies the location of a particular file.
port	A physical connection between the device controller and a peripheral device such as a terminal, printer, or modem.
power cycling	Turning the system power off and then on again.
preen	To run <code>fsck</code> with the <code>-o p</code> option, which automatically fixes any basic file system inconsistencies normally found when a system halts abruptly, without trying to repair more serious errors.

Printer Manager

A tool accessed from the Administration Tool, which is used to add printers to both print servers and print clients. The Printer Manager automatically updates the LP system files and the Service Access Facility files required to configure port monitors for printing.

process

A program in operation.

PROM

Programmable read-only memory. A chip containing permanent, nonvolatile memory and a limited set of commands used to test the system and start the boot process.

request for comments (RFC)

The document series, begun in 1969, which describes the Internet suite of protocols and related experiments. Not all (in fact very few) RFCs describe Internet standards, but all Internet standards are written up as RFCs.

RFC

See **request for comments**.

run level

See **init state**.

server

A system that provides network service such as disk storage and file transfer, or a program that provides such a service.

shell

The command interpreter for a user, specified in the passwd database. The SunOS system software supports the Bourne (default), C, and Korn shells.

slice

An alternate name for a partition. See **partition**.

soft limit

For disk quotas, a limit on file system resources (blocks and inodes) that users can temporarily exceed. Exceeding the soft limit starts a timer. When users exceed the soft limit for the specified time period (default one week), no further system resources are allocated until the user reduces file system use to below the soft limit.

spooling directory	A directory where files are stored until they are processed.
spooling space	The amount of space allocated on a print server for storing requests in the printer queue.
standalone system	A system that has a local disk and can boot without relying on a server.
state flag	A flag in the superblock that the <code>fsck</code> file system check program updates to record the condition of a file system. If a file system state flag is clean, the <code>fsck</code> program is not run on that file system.
striping	Combining one or more physical disks (or disk partitions) into a single logical disk. The logical disk is viewed by the operating system like any other disk-based file system.
superblock	A block on a disk that contains information about a file system, such as its name, size in blocks, and so on. Each file system has its own superblock.
superuser	A user with special privileges granted if the correct password is supplied when logging in as root or using the <code>su</code> command. For example, only the superuser can edit major administrative files in the <code>/etc</code> directory.
swap file	A disk partition or file used to temporarily hold the contents of a memory area until it can be loaded back into memory.
symbolic link	A file that contains a pointer to the name of another file.
system	A computer with a keyboard and terminal. A system can have either local or remote disks, and may have additional peripheral devices such as CD-ROM players, tape drives, diskette drives, and printers.
UID number	The user identification number assigned to each login name. UID numbers are used by the system to identify, by number, the owners of files and directories.

UFS

UNIX file system. The default disk-based file system for the SunOS operating system.

user account

An account set up for an individual user in the passwd database that specifies the user's login name, UID, GID, login directory, and login shell.

user mask

The setting that controls default file permissions assigned when a file or directory is created. The `umask` command controls the user mask settings.

virtual memory

A memory management technique used by the operating system for programs that require more space in memory than can be allotted to them. The kernel moves only pages of the program currently needed into memory, while unneeded pages remain on the disk.

zombie

A process that has terminated but remains in the process table because its parent process has not sent the proper exit code. Zombie processes are removed from the process table when a system is rebooted and do not consume any system resources.

Index

Symbols

- * metacharacter, 214
- ? metacharacter, 214

Numerics

- 4.3 Tahoe file system, 3

A

- accessing remote drive, 129
- adding
 - swap to `vfstab`, 163
- administering quotas, 194, 201
- administrative commands, file systems, 7
- allocated inodes, 222
- appending
 - files to a diskette (`tar`), 74
 - files to a tape, 68
- Archive tape drive, 124
- archives, creating compatible (`cpio`), 77
- archiving files to multiple diskettes (`cpio`), 76
- arguments to mount command, 89
- `autofs`, 21, 29
- `autofs` maps, 29
- automatic file sharing, 55

- `automount` /home, 29
- available disk space
 - monitoring, 171

B

- back file system, 80
- back files, 80
- `backfstype`
 - mount argument, 89
- `backpath`
 - mount argument, 89
- backup
 - checking file systems, 110
 - choosing file systems to, 99
 - combining, 111
 - device names, 103
 - full, 112
 - incremental, 112, 128
 - individual files and directories (`ufsdump`), 128
 - level 0, 112, 126
 - levels 1 through 9, 112
 - media, choosing, 102
 - nine-to-five schedule, 113
 - number of tapes, 125
 - preparing for, 121
 - reasons for, 95
 - record of incremental, 98

- remote systems, 109, 130
- root schedule, 116
- saving tapes, 109
- scheduling, 112
- security issues, 112
- standalone files, 100
- troubleshooting, 130
- using remote drive, 109, 128
- when to run, 108

backup schedule

- choosing, 115
- for root, 116
- nine-to-two-three-four-five, 114
- three-four-five-six-to-two, 115

backup, server

- server files, 100

bad block numbers, 223

bad inode number, 225

bad superblock

- restoring, 258

bad superblock, restoring, restoring

- bad superblock, 258

bar command

- See also* `cpio` command
- header format (`cpio`), 65
- retrieving files created with, 77

`bcheckrc` command

- use of `fsck` state flags, 218

block copy of file system, 59

block size, 20

blocks

- bad, 223
- boot, 19, 270, 271
- data, 19
- directory data, 224
- displaying, 174
- displaying total allocated (`df`), 172
- displaying used, 173
- duplicate, 223
- free, 273
- indirect, 224
- logical size, 275
- regular data, 226
- size, 20

- special inodes, 222
- storage, 19, 273
- storage or data, 270
- superblock, 19

boot block, 19, 270, 271

- installing, 37

booting

- modifying automatic checking, 253, 255

Bourne shell

- setting `PATH` variable, 211
- sourcing dot files, 211

BSD Fat Fast File system, 3

bytes, number per inode, 278

C

C shell

- setting `path` variable for, 211
- sourcing dot files, 211

cached file systems, 79

- back file system, 80
- checking (`fsck`), 91
- concepts, 79
- creating, 81
- deleting, 86
- displaying information about, 85
- front file system, 80
- `maxblocks` parameter, 82
- `maxfiles` parameter, 82
- `minblocks` parameter, 82
- `minfiles` parameter, 83
- modifying parameters, 84
- mount arguments, 89
- mounting, 86
- mounting a CD-ROM, 87
- parameters, 81
- setting parameters, 83
- setting up, 80
- `threshblocks` parameter, 82
- `vfstab` entries, 91

`cachedir`

- mount argument, 89

`cacheid`

- mount argument, 89

capacity of tapes, 137
 cartridge tape
 retensioning, 65
 status, 66
 causes of file system damage, 217
 CD-ROM file systems, mounting, 51
 cfsadmin command, 83
 cfsadmin command, 81, 84, 86
 character special inodes, 222
 characters, wildcard, 214
 checking
 file size, 177, 178
 file system size, 221
 file systems, 255
 file systems interactively, 256
 format and type of inodes, 222
 free blocks, 221
 free inodes, 221
 inode list for consistency, 221
 modifying automatic boot, 253
 quota consistency, 203
 chgrp command, 215
 chmod command, 214
 chmod table of options, 214
 chown command, 213
 clearing /var/tmp, 189
 clri command, 7
 codes used by fuser, 53
 command line, mounting from, 47
 Command not found error message, 209
 commands
 executing, 212
 file system-specific, 7
 for monitoring files and
 directories, 176
 generic file system, 7
 swap, 167, 168
 table of disk quota, 194
 compatible archives, creating (cpio), 77
 configuring file systems for disk
 quotas, 198
 consistency, checking quota, 203
 controlling disk space with quotas, 193

 copying
 all files in a directory to tape, 70
 complete file systems (dd), 63
 data using ufsdump, 98
 directories between file systems, 64
 file systems (volcopy), 61
 files, 58
 files to a diskette (tar), 73
 files to diskette, 72
 files to tape (tar), 67
 files to tape (tar, cpio), 65
 files with different header format
 (cpio -H), 76
 groups of files, 64
 individual files, 64
 core files, 189
 cpio command, 58, 64, 65, 69, 76
 crash dumps
 deleting, 190
 removing, 190
 crc header format (cpio), 65
 creating
 boot block, 37
 compatible archives (cpio), 77
 entries in file system table, 42
 file system on diskette, 37
 file systems, 35
 hard links, 191
 loopback file system, 39
 mount point, 22, 42
 new file systems, 20
 swap file, 166
 symbolic links, 191
 temporary file system, 38
 UFS file system, 36
 custom parameters for file systems, 274
 cylinder groups, 19

D
 damage to file systems, 217
 data block, 19, 226, 270
 data directory blocks, 224
 dd command, 58, 63

default
 / and /usr directories, 265, ?? to 270
 file system for /tmp (TMPFS), 4
 mount options, 47, 52
 quota time limit, changing, 198
 SunOS file system, 12
 tape drive densities, 104

delay, rotational, 20, 277

deleting
 cached file systems, 86
 core files, 189, 190
 crash dumps, 190
 list of inactive files, 187
 temporary files, 188

densities, specifying tape drive, 104

detecting end-of-media (ufsdump), 97, 137

determining
 file system types, 9
 restore tapes, 140
 tape device name, 142
 type of tape drive, 142

device names
 backup, 103
 command (devnm), 10
 disk, 13
 diskette drive, 103
 finding disk, 141
 finding raw, 122
 finding tape, 142
 table of tape controller, 105
 tape drive, 103

device to fsck field (vfstab), 27

device to mount field (vfstab), 27

devices
 for disk-based file systems, 3

devnm command, 10, 122

/dev/rmt directory, 103

df command, 7, 172
 options, 172

dfstab file, 30, 55

directories, 213
 /dev/rmt, 103
 /proc, 5
 /tmp, 4
 /var/crash, 190

backing up individual
 (ufsdump), 128

changing permissions, 213

commands for monitoring, 176

copying between file systems, 64

copying files to tape, 70

default / and /usr, 265 to 270

disconnected, 226

displaying size, 181

displaying size of, 182

finding large, 180

inodes, 222

unallocated blocks, 225

disabling
 quotas for individual users, 203

disconnected directories, 226

disk device name, finding, 141

disk names
 understanding, 13

disk partition information, displaying
 (df), 172

disk quotas, 195
 administering, 201
 configuring file systems for, 198
 table of commands, 194

disk slice, 14
 formatting, 14
 getting information, 15

disk space
 controlling with quotas, 193
 displaying
 percent capacity, 174
 used, 174
 finding large users of, 182
 monitoring, 171

disk use
 finding (du), 180
 reporting on, 204

disk-based file systems, 3

diskette
 appending files to (tar), 74
 archiving files to multiple (cpio), 76

- copying files to, 72
- copying files to (tar), 73
- creating file system on, 37
- ejecting, 72
- listing files on (tar), 74
- retrieving files from (tar), 75

diskette drive, device name, 103

displaying

- blocks and files, 174
- current search path, 210
- percent capacity of disk, 174
- size of directories, 181, 182
- swap space, 165
- total blocks allocated (df), 172
- used blocks and files, 173
- used disk space, 174
- user allocation of space, 183

doing incremental backups, 128

DOS file system, 3

dot files

- sourcing for Bourne shell, 211
- sourcing for C shell, 211
- sourcing for Korn shell, 211

du command, 180

dump command

- See also* `ufsdump` command
- incompatibility, 151

dumpdates file, 98

duplicate blocks, 223

E

echo \$PATH command, 210

editing

- quotas, 196

edquota command, 196, 198

ejecting a diskette, 72

Emulex MT-02 tape drive, 124

end-of-media detection (`ufsdump`), 97, 137

end-of-media recognition (`cpio`), 64

entries, creating in file system table, 42

error messages

- `fsck`, 226 to 253

Exabyte tape drive, 124

example

- backup strategy for a server, 117
- copying file systems (`volcopy`), 61
- creating a `ufs` file system, 37
- finding large files, 179
- format command, 16
- full backup, 127
- number of tapes for backup, 125
- restoring bad superblock, 259
- restoring complete file system, 144
- restoring files interactively, 148
- `tar`, 67
- `vfstab` file entries, 43

executing a command, 212

/export/home directory, 12

exporting files, *See* sharing

extended fundamental types (UFS file system), 13

F

FDFS file system, 6

ff command, 7

FIFO inodes, 222

FIFOFS file system, 6

file size, maximum, 18

file system table

- creating entries, 42
- virtual, 26

file systems

- /, 12
- /export/home, 12
- /opt, 13
- /proc, 13
- /usr, 12
- /var, 12

4.3 Tahoe, 3

administrative commands, 7

BSD Fat Fast, 3

cached, 79

checking, 255

checking before backup, 110

checking interactively, 255, 256
 checking size, 221
 configuring for disk quotas, 198
 copying (`volcopy`), 61
 copying complete (`dd`), 63
 creating loopback, 39
 creating new, 20
 creating on diskette, 37
 creating UFS, 36
 custom parameters, 274
 cylinder group struct, 270
 damage to, 217
 default SunOS, 12
 definition of, 1
 devices for disk-based, 3
 disk-based, 3
 DOS, 3
 FDFS, 6
 FIFOFS, 6
 finding mounted, 41
 finding raw device names, 122
 finding types, 9
 finding types (`fstyp`), 11
 finding users, 184
 fixing, 260
 High Sierra, 3
 ISO 9660, 3
 labeling, 59
 literal (block) copy, 59
 LOFS, 5, 39
 loopback, 5
 making available, 21
 manual pages, 9
 mount point, 8
 mount table, 25
 mounting, 21
 mounting by type, 47
 mounting NFS, 49
 moving directory trees between, 192
 NAMEFS, 6
 network-based, 3
 PC, 3
 planning UFS, 19
 preening, 257, 258
 process, 5
 PROCFS, 5
 pseudo, 4
 reasons for inconsistencies, 219
 reducing overloaded, 186
 requirements for mounting, 22
 restoring complete, 142, 143
 root full, 130
 sharing, 21, 29, 55
 SPECFS, 7
 special device, 8
 specific commands, 7
 SWAPFS, 7
 syntax of generic commands, 8
 table (`vfstab`), 21, 26
 table of administrative commands, 7
 terminating all processes, 53
 TMPFS, 4
 total size of UFS, 18
 types of, 2, 8
 UFS, 3
 understanding UFS, 13
 UNIX, 3
 unmounting, 25, 53, 54
 which to back up, 99
 why you back up, 95

files
 /etc/default/fs, 9
 /etc/dfs/fstypes, 9
 appending to a diskette (`tar`), 74
 appending to tape, 68
 archiving to multiple diskettes
 (`cpio`), 76
 available in file system, 172
 back up, 100
 back up, standalone, 100
 changing ownership, 213
 changing permissions, 213
 checking size, 177, 178
 commands for monitoring, 176
 copying all to tape, 70
 copying groups of, 64
 copying individual, 64
 copying to diskette, 72, 73
 copying to tape (`tar`), 67
 deleting list of inactive, 187
 deleting old or inactive, 187

displaying for UFS file system, 174
 displaying used, 173
 finding inactive, 185
 finding large, 178
 finding old, 185
 for setting search path, 210
 in the `/proc` directory, 6
 linking, 191
 listing on diskette (`tar`), 74
 listing on tape, 67, 70
 maximum number of, 18
`mnttab`, 10
 monitoring log, 176
 number in file system, 172
 number of, 278
 restoring interactively, 147
 restoring specific, 149
 retrieving from diskette (`tar`), 75
 retrieving from tape, 69, 71
 sharing, 29, 55
 that grow, 176
 truncating growing, 186
`vmcore.n`, 190
 filesystem type, determining, 9
`find` command, 185
 finding
 all tape drives, 125
 core files, 190
 current search path, 210
 disk device name, 141
 disk slice information, 15
 large directories, 180
 large files, 178
 large space users, 182
 mounted file systems, 41
 number of tapes for backup, 125
 old files, 185
 problems with search paths, 209
 raw device names for file
 systems, 122
 restore tapes, 140
 tape device name, 142
 tape drive type, 124
 type of file system, 9
 user allocation of space, 183
 users of file system space, 184
 fixing bad file systems, 260
 floppy diskette, *See* diskette
`format` command, 15, 16
 format of inodes, checking, 222
 formats, header (`cpio`), 65
 formatting
 disk slice, 14
 fragment size, 19, 20, 275
 free blocks, 273
 checking, 221
 free inodes, checking, 221
 free space, minimum, 20, 276
 front file system, 80
`fs` file, 9
`FS` type field (`vfstab`), 27
`FSACTIVE` state flag (`fsck`), 218
`FSBAD` state flag (`fsck`), 219
`fsck` command, 7, 218
 checking free blocks, 221
 checking free inodes, 221
 checking inode list size, 221
 checking superblock, 220
 conditions to repair, 219
 error messages, 226 to 253
 `FSACTIVE` state flag, 218
 `FSBAD` state flag, 219
 `FSCLEAN` state flag, 218
 `FSSTABLE` state flag, 218
 preening, 257
 state flags, 218
 syntax and options, 260 to 263
 table of state flag conditions, 219
 using interactively, 255
`fsck` pass field (`vfstab`), 27, 253
`FSCLEAN` state flag (`fsck`), 218
`fsdb` command, 7
`FSSTABLE` state flag (`fsck`), 218
`fstab` file, *See* `vfstab` file
`fstyp` command, 7, 11
`fstypes` file, 9
 full backup, 112

full root file system, 130

fuser command, 53

G

gap, *See* rotational delay

generic file system commands, 7

grep command, 9

group ownership
changing for a file, 215

groups, cylinder, 19

growing files, truncating, 186

H

hard limit (quotas), 193

hard links, 191

header format

copying files with different
(cpio -H), 76

cpio, 65

crc, 65

odc, 65

tar, 65

ustar, 65

High Sierra file system, 3

High Sierra file systems

mounting, 51

/home, automounted, 29

how many tapes for backup, 125

HSFS, *See* High Sierra file system

I

inactive files

deleting, 187

finding, 185

inconsistencies in file systems, 219

incorrect . and .. entries, 225

incremental backup, 98, 112, 128

indirect blocks, 224

information

disk slice, 15

initializing quotas, 200

inode list size, 221

inode states

allocated, 222

partially allocated, 222

unallocated, 222

inodes, 19, 270, 272

bad number, 225

block special, 222

character special, 222

checking format and type, 222

directory, 222

FIFO, 222

link count, 222

number of, 20

number of bytes per, 278

regular, 222

size, 224

symbolic link, 222

installing boot block, 37

interactive

checking file systems, 255, 256

commands for restore, 154

restore, 147

ISO 9660 file system, 3

K

keeping backup tapes, 109

keyword=value pairs, 63

killing all processes for a file system, 53

Korn shell

setting PATH variable, 211

sourcing dot files, 211

whence command, 212

L

labeling file systems, 59

labelit command, 7, 58, 59, 60

level 0 backup, 112, 126

levels 1 through 9 backup, 112

limit

hard (quotas), 193

- size (UFS file system), 18
- soft (quotas), 193
- link count of inodes, 222
- links, 191
- listing
 - files on a diskette (`tar`), 74
 - files on a tape, 67, 70
- `ln` command, 191
- local file systems, mounting, 46
- `local-accessa`
 - mount command argument, 90
- LOFS file systems, 5, 39
- log
 - record of dumps, 98
- log files
 - list of, 176
 - monitoring, 176
- logical block size, 19, 275
- loopback file system, 5
 - creating, 39
 - mounting, 43
- `lost+found` directory, 218

M

- magnetic tape cartridge
 - retensioning, 65
 - rewinding, 66
 - status, 66
- maintaining tape drives, 107
- manual pages, file system commands, 9
- maps
 - `autofs`, 29
- `maxblocks`
 - `cfsadmin` parameter, 82
- `maxfiles`
 - `cfsadmin` parameter, 82
- maximum file size, 18
- maximum number of files, 18
- media, choosing backup, 102
- memory storage, virtual, 160
- metacharacters, 214
 - using with `tar` command, 67
- `minblocks`
 - `cfsadmin` parameter, 82
- `minfiles`
 - `cfsadmin` parameter, 83
- minimum free space, 20, 276
- `mkfile` command, 166, 167
- `mkfs` command, 7, 35, 36
- `mntopts` field of `/etc/mnttab`, 196
- `mnttab` file, 10, 25
- modifying automatic boot checking, 253, 255
- monitoring
 - disk space, 171
 - files and directories, commands, 176
 - log files, 176
- `mount` command, 7, 10, 28, 42
- `mount` command arguments, 89
- `mount` options field (`vfstab`), 28
- mount point, 8
 - creating, 22, 42
- `mount` point field (`vfstab`), 27
- mount table, 25
- `mountall` command, 7, 28, 45
- mounted file systems, finding, 41
- mounting
 - all files in `vfstab` file, 45
 - CD-ROM file systems, 51
 - file systems, 21
 - file systems automatically, 29
 - file systems by type, 47
 - from a command line, 47
 - High Sierra file systems, 51
 - local file systems, 46
 - LOFS file systems, 43
 - NFS file systems, 43, 49
 - PCFS file systems, 52
 - remote file systems, 46
 - requirements, 22
 - `s5fs` file systems, 52
 - table of options `nfs`, 50
 - table of options `UFS`, 49
 - UFS file systems, 43
 - using default options, 47, 52

mounting a CD-ROM as a cached file system, 87
mounting cached file systems, 86
mounting file systems, 22
moving directory trees between file systems, 192
mt command, 65, 124

N

NAMEFS file system, 6
ncheck command, 7
network
 recognizing access problems, 215
network-based file systems, 3
new file systems, creating, 20
newfs command, 35, 36, 278
 parameters, 20
NFS, 29
NFS file systems
 mounting, 43
NFS server, definition of, 30
nine-to-five backup schedule, 113
nine-to-two-three-four-five backup schedule, 114
noconst
 mount command argument, 90
non-shared
 mount command argument, 90

O

odc header format (cpio), 65
old files, 185
 deleting, 187
 finding, 185
optimization type, 20
 space, 277
 time, 277
options
 for du, 180
 for mkfile, 167
 for ufsrestore, 152

mount, table of NFS, 50
mount, table of UFS, 49
tape, no rewind, 104
tape, rewind, 104
 ufsdump command, 131
overloaded file systems, reducing, 186
ownership
 changing, 213
 changing file group, 215
 troubleshooting problems with, 212
 why it changes, 213

P

parameters
 file system custom, 274
 UFS file system, 20
parameters used by newfs, 20
partially allocated inodes, 222
partition
 displaying information (df), 172
 swap, 160
partition command, 15
path
 displaying current, 210
 recognizing problems with, 209
 troubleshooting, 210
 verifying search, 211
PC file system, 3
PCFS file system
 mounting, 52
percent capacity, displaying disk, 174
permissions
 changing, 213
 read/write, 214
 troubleshooting problems with, 212
planning
 backup schedule, 112
 swap area, 20
 UFS file systems, 19
polling system for tape drives, 125
preening file systems, 257, 258
preparing for backup, 121
preparing to restore, 139

print volume table of contents, 17
problems with permission and
ownership, 212
problems with search paths, 209
/proc directory, 5, 13
process file system, 5
PROCFS file system, 5
prototype quotas, using, 200
prtvtoC command, 17
pseudo file systems, 4

Q

QIC format tape drive, 107
QIC-150 tape drive, 124
QIC-24 tape drive, 124
quot command, 182, 205
quota command, 204
quotacheck command, 200, 204
quotaoff command, 196, 202
quotaon command, 196, 201
quotas, 193
administering, 194, 201
changing default time limit, 198
changing for individual users, 203
changing soft time limits, 202
checking consistency, 203
configuring file systems for, 198
disabling for individual users, 203
editing, 196
hard limit, 193
how they work, 195
how users are affected, 195
initializing, 200
reporting on, 204
setting up, 194
setting up user, 199
soft limit, 193
table of commands, 194
turning off, 202
turning on, 201
using prototype, 200
what they do, 193

quotas file, 195, 198

R

rack-mounted tape drives, 106
rdist command, 58
read/write permissions, 214
recognizing network access problems, 215
record of dumps, 98
record of incremental backup, 98
recylinder group, structure of, 270
reducing overloaded file systems, 186
regular inodes, 222
remote drive
accessing, 129
restoring from, 150
using for backups, 128
remote file systems, mounting, 46
remote systems, backup, 130
removing
core files, 189
crash dumps, 190
old or inactive files, 187
swap file from use, 168
reporting on quotas, 204
repquota command, 204
requirements for mounting a file
system, 22
restore
complete file system, 143
finding tapes, 140
from remote drive, 150
interactive commands, 154
interactively, 147
preparing to, 139
specific files, 149
type of tape drive, 142
using matching commands, 150
restore command, *See* ufsrestore
command
restoring bad superblock, 258
retensioning magnetic tape cartridge, 65
retrieving

- files created with `bar` command, 77
- files from a tape, 69, 71
- files from diskette (`tar`), 75
- files from tape, 71
- rewinding magnetic tape cartridge, 66
- `rm` command, 187
- Rock Ridge extension (HSFS file system), 3
- root (`/`) file system, 12
- root file system
 - full, 130
- rotational delay, 20, 277

S

- saving backup tapes, 109
- scheduling backups, 108, 112
- SCSI tape drives, 107
 - maximum number, 105
- search path
 - displaying current, 210
 - files for setting, 210
 - recognizing problems with, 209
 - troubleshooting, 210
 - verifying, 211
- security issues, backup, 112
- server files to back up, 100
- setting
 - `PATH` variable for Bourne shell, 211
 - `path` variable for C shell, 211
 - `PATH` variable for Korn shell, 211
 - size limit on temporary file system, 38
- setting up
 - quotas, 194
 - user quotas, 199
- `share` command, 30
- `shareall` command, 30
- sharing
 - file systems, 21
 - files, 29, 30, 55
- showing total blocks allocated (`df`), 172
- size
 - checking file, 177, 178
 - checking file system, 221
 - displaying directory, 181, 182
 - fragment, 19, 275
 - inode, 224
 - logical block, 19
- size limit, temporary file system, 38
- size of UFS file systems, 18
- size restrictions (UFS file system), 18
- slice
 - disk, 14
 - formatting, 14
- soft limit (quotas), 193
- `source` command, 211
- sourcing
 - Bourne shell dot files, 211
 - C shell dot files, 211
 - Korn shell dot files, 211
- space
 - displaying user allocation, 183
 - finding large users of, 182
 - optimization type, 277
- SPECFS file system, 7
- special device, 8
- specifying tape drive densities, 104
- standalone, files to back up, 100
- state flag
 - `fsck`, 218
 - table of `fsck` conditions, 219
 - UFS file systems, 13
- status
 - magnetic tape cartridge, 66
- stopping
 - all processes for a file system, 53
- storage block, 19, 270, 273
- storage capacities, media, 102
- storage, virtual memory, 160
- streaming tape commands, 65
- structure of cylinder groups, 270
- SunOS default file system, 12
- superblock, 19, 220, 270, 271
 - restoring bad, 258

swap
 displaying existing, 165
 swap area, 20
 swap command, 166, 167, 168
 swap file
 adding to `vfstab`, 163
 creating, 166
 removing from use, 168
 swap partition, 160
 swapadd command, 163
 SWAPFS file system, 7
 symbolic link, 191
 creating, 191
 inodes, 222
 syntax
 dd command, 63
 fsck command, 260 to 263
 generic file system commands, 8
 newfs, 278
 remote drive backup, 109
 Sysgen tape drive, 124
 system
 finding tape drives, 125

T

table
 commands for copying files, 58
 default SunOS file system, 12
 disk quota commands, 194
 file system administrative
 commands, 7
 fsck state flag conditions, 219
 fuser codes, 53
 interactive commands for
 ufsrestore, 154
 media storage capacities, 102
 mount, 25
 mount options NFS, 50
 mount options UFS, 49
 options for `df`, 172
 options for `du`, 180
 options for `mkfile`, 167
 options for `ufsdump`, 132
 options for `ufsrestore`, 152
 SCSI tape drive density, 107
 tape capacity, 137
 tape controller device names, 105
 tail command, 186
 tape
 appending files to, 68
 capacity, 137
 characteristics, 137
 combining backup, 111
 commands, 65
 copying all files in a directory, 70
 copying files to (`tar`, `cpio`), 65
 listing files on, 67, 70
 no rewind option, 104
 retrieving files from, 69, 71
 rewind option, 104
 sizes, 102
 storage capacities, 102
 tape controller, table of device names, 105
 tape drive
 Archive, 124
 default densities, 104
 determining type for restore, 142
 device names, 103
 Emulex MT-02, 124
 Exabyte, 124
 finding all, 125
 finding type, 124
 maintaining, 107
 maximum SCSI, 105
 QIC format, 107
 QIC-150, 124
 QIC-24, 124
 rack-mounted, 106
 restoring from remote, 150
 SCSI, 107
 specifying densities, 104
 Sysgen, 124
 Wangtek, 124
 Xylogics 472, 124
 tape, magnetic cartridge, 66
 retensioning, 65
 status, 66
 tar command, 58, 65, 67, 74

tar header format (`cpio`), 65
temporary file system, 4
 creating, 38
 setting size limit, 38
temporary files, deleting, 188
terminating all processes, 53
three-four-five-six-to-two backup
 schedule, 115
threshblocks
 cfsadmin parameter, 82
time limit
 changing quota default, 198
 soft, 202
time, optimization type, 277
/tmp directory, 13
/tmp directory, 4
TMPFS file system, 4
troubleshooting
 backup, 130
 permission and ownership
 problems, 212
 problems with search paths, 210
truncating files that grow, 186
turning off quotas, 202
turning on quotas, 201
type of file systems, 2, 8
 mounting by, 47
type of inodes, checking, 222
type of tape drive, finding, 124

U

ufs command
 displaying blocks and files, 174
UFS file system, 3, 13
 cylinder groups, 19
 extended fundamental types, 13
 fragment size, 19
 large file systems, 13
 logical block size, 19
 maximum file size, 18
 maximum number of files, 18
 mounting, 43

 parameters, 20
 size restrictions, 18
 state flags, 13
 total size, 18
ufsdump command, 96, 125
 advantages, 96
 copying data with, 98
 disadvantages, 97
 end-of-media detection, 97
 how it works, 97
 options and arguments, 131
ufsrestore command, 139, 143
 options and arguments, 151
umount command, 7, 28, 54
umountall command, 7, 28
unallocated directory blocks, 225
unallocated inodes, 222
UNIX file system, 3
unmounting file systems, 22, 25, 53, 54, 55
used disk space, displaying, 174
user quotas, setting up, 199
users
 affected by quotas, 195
 changing quotas for, 203
 disabling quotas for, 203
 displaying allocation of space, 183
 displaying file system, 184
 finding large space, 182
 setting up quotas with prototype, 200
/usr file system, 12
ustar header format (`cpio`), 65

V

/opt directory, 13
/var directory, 12
/var/crash directory, 190
/var/tmp directory, clearing, 189
verifying the search path, 211
vfstab file, 9, 21, 22, 122, 163
 adding swap to, 163
 creating entries in, 42
 default, 26

- device to fsck field, 27
- device to mount field, 27
- editing for quotas, 198
- entry for loopback file system, 39
- entry for temporary file system, 38
- FS type field, 27
- fsck pass field, 27
- modifying fsck pass, 253
- mount at boot field, 28
- mount options field, 28
- mount point field, 27
- mounting all files, 45
- vfstab), 28
- virtual file system table, 26
- virtual memory storage, 160
- vmcore.*n* file, 190
- volcopy command, 7, 58, 59, 61
- volume name
 - assigning (labelit), 60
 - finding (labelit), 60
- volume table of contents, print, 17

W

- Wangtek tape drive, 124
- whence command, 212
- which command, 211, 212
- wildcard characters, 214
- write-around
 - mount command argument, 90

X

- Xylogics 472 tape drive, 124

