



A/UX Local System Administration

Release 3.0

LIMITED WARRANTY ON MEDIA AND REPLACEMENT

If you discover physical defects in the manuals distributed with an Apple product or in the media on which a software product is distributed, Apple will replace the media or manuals at no charge to you, provided you return the item to be replaced with proof of purchase to Apple or an authorized Apple dealer during the 90-day period after you purchased the software. In addition, Apple will replace damaged software media and manuals for as long as the software product is included in Apple's Media Exchange Program. While not an upgrade or update method, this program offers additional protection for up to two years or more from the date of your original purchase. See your authorized Apple dealer for program coverage and details. In some countries the replacement period may be different; check with your authorized Apple dealer.

ALL IMPLIED WARRANTIES ON THE MEDIA AND MANUALS, INCLUDING IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, ARE LIMITED IN DURATION TO NINETY (90) DAYS FROM THE DATE OF THE ORIGINAL RETAIL PURCHASE OF THIS PRODUCT.

Even though Apple has tested the software and reviewed the documentation, **APPLE MAKES NO WARRANTY OR REPRESENTATION, EITHER EXPRESS, OR IMPLIED, WITH RESPECT TO SOFTWARE, ITS QUALITY, PERFORMANCE, MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE. AS A RESULT, THIS SOFTWARE IS SOLD "AS IS," AND YOU, THE PURCHASER, ARE ASSUMING THE ENTIRE RISK AS TO ITS QUALITY AND PERFORMANCE.**

IN NO EVENT WILL APPLE BE LIABLE FOR DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES RESULTING FROM ANY DEFECT IN THE SOFTWARE OR ITS DOCUMENTATION, even if advised of the possibility of such damages. In particular, Apple shall have no liability for any programs or data stored in or used with Apple products, including the costs of recovering such programs or data.

THE WARRANTY AND REMEDIES SET FORTH ABOVE ARE EXCLUSIVE AND IN LIEU OF ALL OTHERS, ORAL OR WRITTEN, EXPRESS, OR IMPLIED. No Apple dealer, agent, or employee is authorized to make any modification, extension, or addition to this warranty.

Some states do not allow the exclusion or limitation of implied warranties or liability for incidental or consequential damages, so the above limitation or exclusion may not apply to you. This warranty gives you specific legal rights, and you may also have other rights which vary from state to state.



Apple Computer, Inc.

This manual and the software described in it are copyrighted, with all rights reserved. Under the copyright laws, this manual or the software may not be copied, in whole or part, without written consent of Apple, except in the normal use of the software or to make a backup copy of the software. The same proprietary and copyright notices must be affixed to any permitted copies as were affixed to the original. This exception does not allow copies to be made for others, whether or not sold, but all of the material purchased (with all backup copies) may be sold, given, or loaned to another person. Under the law, copying includes translating into another language or format.

You may use the software on any computer owned by you, but extra copies cannot be made for this purpose.

The Apple logo is a registered trademark of Apple Computer, Inc. Use of the “keyboard” Apple logo (Option-Shift-K) for commercial purposes without the prior written consent of Apple may constitute trademark infringement and unfair competition in violation of federal and state laws.

© Apple Computer, Inc., 1992
20525 Mariani Avenue
Cupertino, CA 95014
(408) 996-1010

Apple, the Apple logo, AppleTalk, A/UX, EtherTalk, ImageWriter, LaserWriter, LocalTalk, Macintosh, MacTerminal, and MPW are trademarks of Apple Computer, Inc., registered in the United States and other countries.

Finder and MacX are trademarks of Apple Computer, Inc.

Adobe, Adobe Illustrator, and PostScript are registered trademarks of Adobe Systems Incorporated.

Electrocomp 2000 is a trademark of Image Graphics, Inc.

ITC Garamond and ITC Zapf Dingbats are registered trademarks of International Typeface Corp.

Linotronic is a registered trademark of Linotype Co.

MacWrite and MacPaint are trademarks of Claris Corporation.

Microsoft is a registered trademark of Microsoft Corp.

NFS is a trademark of Sun Microsystems, Inc.

NuBus is a trademark of Texas Instruments.

PostScript and TranScript are registered trademarks of Adobe Systems, Incorporated.

QuarkXPress is a registered trademark of Quark, Inc.

UNIX is a registered trademark of UNIX System Laboratories.

VT100 and VT102 are trademarks of Digital Equipment Corporation.

X-Window System is a trademark of Massachusetts Institute of Technology.

Yellow Pages is a registered trademark of British Telecommunications plc.

Simultaneously published in the United States and Canada.

Mention of third-party products is for informational purposes only and constitutes neither an endorsement nor a recommendation. Apple assumes no responsibility with regard to the performance or use of these products.

Contents

Figures and tables / xvii

About this Guide / xix

Who this manual is for / xix

What you need to know / xx

What this manual contains / xx

Conventions used in this guide / xxii

Keys and key combinations / xxii

Terminology / xxiii

The `Courier` font / xxiii

Font styles / xxiv

A/UX command syntax / xxiv

Manual page reference notation / xxv

For more information / xxvi

1 Concepts of System Administration / 1-1

Some suggestions before you begin / 1-3

Customizing your system / 1-5

Naming the system / 1-5

Changing the message of the day / 1-6

Setting the system time / 1-6

Resetting after moving a system to a different time zone / 1-7

- Administrative logins and groups / 1-7
 - Administrative logins / 1-8
 - Administrative groups / 1-9
- Single-user and multi-user states / 1-9
 - Single-user state / 1-10
 - Multi-user state / 1-11
- Initial processes: `/etc/inittab` / 1-11
 - Changing run levels: `init` / 1-12
- Configuring the kernel / 1-13
 - Autoconfiguration / 1-14
 - The `newconfig` program / 1-15
 - Changing kernel parameters / 1-17
- Two types of UNIX file systems / 1-17
- About A/UX files / 1-19

2 System Startup and Shutdown / 2-1

- Starting up the system / 2-4
- About A/UX Startup / 2-5
 - A/UX Startup command shell window / 2-5
 - Commands that run in A/UX Startup / 2-6
 - A/UX Startup menus / 2-8
 - Apple menu / 2-8
 - File menu / 2-8
 - Edit menu / 2-9
 - Execute menu / 2-9
 - Preferences menu / 2-10
 - The startup sequence / 2-13
 - Checking / 2-13
 - Loading / 2-14
 - Launching / 2-14
 - Checking root file system / 2-15
 - Initializing device drivers / 2-15
 - Checking file systems / 2-16
 - Starting background processes / 2-17
- Logging out, restarting, and shutting down / 2-18
 - Logging out from the command line / 2-18
 - Entering single-user state from multi-user state / 2-18
 - Using the `shutdown` command / 2-19
 - The natural order of startup devices / 2-21
- Changing the A/UX startup device / 2-22

3 User and Group Administration / 3-1

- The A/UX working environment / 3-3
 - An introduction to the working environment / 3-3
 - Command shells and setup files / 3-6
 - Command line prompt / 3-7
 - The C shell setup files / 3-7
 - The Bourne shell setup files / 3-8
 - The Korn shell setup files / 3-8
 - Macintosh System Folders / 3-8
 - How A/UX establishes the environment / 3-9
- Administering user accounts / 3-10
 - The `/etc/passwd` file / 3-10
 - Format of `/etc/passwd` / 3-10
 - Adding a user / 3-11
 - Planning a user's working environment / 3-12
 - Adding users manually / 3-14
 - Removing users / 3-19
 - Voiding a user account / 3-19
 - Deleting a user account / 3-20
 - Moving a user's home directory within a file system / 3-21
 - Moving a user's directory across file systems / 3-21
 - Using `cpio` / 3-21
 - Using `tar` / 3-22
 - Changing a user's shell / 3-23
- Administering groups / 3-24
 - Creating groups / 3-24
 - The `/etc/group` file / 3-25
 - Multiple group memberships / 3-26
- Files, directories, folders, and permissions / 3-27
 - UNIX file-access permissions / 3-27
 - File permissions / 3-28
 - Directory and folder permissions / 3-29
 - Changing file permissions: `chmod` / 3-29
 - Symbolic terms / 3-30
 - Numeric terms / 3-31
 - Commands that assume permissions using `setuid` and `setgid` / 3-32
 - The sticky bit / 3-34
 - Setting default file permissions with `umask` / 3-34

How UNIX permissions and Macintosh File-Sharing permissions compare / 3-35
Security / 3-38
 Know your open accounts / 3-38
 Monitor `/etc/passwd` and `/etc/group` / 3-38
 Multiple users—one login / 3-39
 Password aging / 3-39
 Password restrictions / 3-39
 Permission restrictions / 3-40

4 Preparing a Hard Disk for A/UX / 4-1

The steps in preparing disks for A/UX / 4-3
About UNIX file systems / 4-3
 Disk partitions / 4-3
 Identifying partitions / 4-5
 File systems / 4-6
 Accessing file systems / 4-7
Background on Apple Hard Disk SC Setup / 4-9
 Choosing a partition / 4-11
Partitioning disks / 4-12
 Considerations before you begin / 4-12
 Removing a partition / 4-13
 Adding a partition / 4-15
 Grouping free space / 4-21
 Moving a partition / 4-23
 Viewing information about partitions / 4-24
 Adding swap space / 4-26
Making A/UX file systems / 4-29
 Disk parameters: `/etc/disktab` / 4-29
 Making Berkeley file systems: `newfs` / 4-30
 Making System V file systems: `mkfs` / 4-31
Mounting A/UX file systems / 4-32
 Mounting automatically: `fsentry` / 4-33
 Mounting temporarily: `mount` / 4-34
 Unmounting file systems: `umount` / 4-36

5	Backing Up Your System / 5-1
	Backup overview / 5-3
	Full versus partial backups / 5-3
	A common backup scheme / 5-3
	Referring to devices by device file names / 5-4
	Multiple archives on a single tape / 5-6
	Run levels and backups / 5-7
	Backing up mounted file systems / 5-7
	Backup media / 5-7
	Storage capacity of backup media / 5-8
	When to use floppy disks / 5-8
	When to use tape / 5-9
	When to use paper / 5-10
	The backup utilities / 5-10
	About the Finder / 5-11
	About <code>cpio</code> / 5-11
	<code>cpio</code> and tape backups / 5-13
	<code>cpio</code> and the A/UX 3.0 Installation CD-ROM / 5-14
	Using <code>cpio</code> / 5-14
	Backing up specific files / 5-15
	Backing up all files in a directory / 5-15
	Creating partial backups / 5-16
	Creating multiple-archive backups / 5-16
	Listing the contents of the backup medium / 5-16
	Extracting specific files from disk or tape / 5-17
	Extracting all files from disk or tape / 5-18
	Extracting files from multiple-archive backups / 5-19
	In the event of hard I/O errors / 5-19
	About <code>tar</code> / 5-20
	<code>tar</code> and tape backups / 5-21
	If a backup requires multiple volumes / 5-21
	Copying to a disk / 5-22
	Using <code>tar</code> / 5-23
	Backing up specific files / 5-23
	Backing up an entire directory / 5-24
	Creating multiple-archive backups / 5-24
	Appending a file to a disk / 5-25
	Adding a later version of a file to a disk / 5-25
	Listing the contents of the backup medium / 5-26
	Extracting a specific file or directory / 5-26
	Extracting files from multiple-archive backups / 5-28
	Extracting a particular version of a file / 5-28

- About `dump.bsd` and `restore` / 5-29
 - Dump levels / 5-30
 - Using dump levels in a monthly backup strategy / 5-30
 - `dump.bsd`, `restore`, and tape backups / 5-31
- Using `dump.bsd` / 5-32
 - Full backups / 5-32
 - Incremental backups / 5-33
 - Finding out which file systems to back up / 5-34
- Using `restore` / 5-34
 - Listing the contents of the backup medium / 5-34
 - Restoring individual files / 5-35
 - Restoring a full backup / 5-36
 - Interactive mode for `restore` / 5-38
- About `pax` / 5-40
 - `pax` and tape backups / 5-40
- Using `pax` / 5-40
 - Backing up specific files / 5-41
 - Backing up all files in a directory / 5-41
 - Listing the contents of the backup medium / 5-42
 - Extracting specific files from a disk or tape / 5-42
 - Extracting all files from a disk or tape / 5-43
- Verifying data on backed-up disks / 5-43
- Automating system administration with `cron` / 5-45
 - User access to `cron` / 5-45
 - The crontab file format / 5-46
 - Adding `cron` commands / 5-47
 - Changing `cron` commands / 5-48
 - Removing `cron` commands / 5-48

6 Managing Disks / 6-1

- About autorecovery / 6-3
 - Overview / 6-3
 - Using autorecovery / 6-4
 - How autorecovery works / 6-5
 - Autorecovery administration / 6-6
 - The `eu` utility / 6-6
 - The `eupdate` utility / 6-7
 - Troubleshooting / 6-7
 - A/UX Startup errors / 6-7
 - Checking the autorecovery file system / 6-8
 - Boot time errors / 6-9
 - Run time errors / 6-10
 - Clearing autorecovery partition names / 6-10

- Reclaiming disk space / 6-11
 - Removing unneeded files / 6-12
 - Trimming files that grow / 6-13
 - Compressing infrequently used files / 6-14
 - Usage notes / 6-15
 - Extensions to names of compressed files / 6-15
 - Compressing an archive of files / 6-16
- CD-ROM and A/UX / 6-16
 - Mounting a CD-ROM as an A/UX file system / 6-17
 - Accessing the man pages on a CD-ROM / 6-17

7 Managing Printers, Terminals and Modems / 7-1

- Peripheral device drivers / 7-3
- Ports / 7-3
- Managing the `lpr` print spooler / 7-4
 - Definitions / 7-5
 - Setting up the print spooler / 7-6
 - The `printcap` database / 7-6
 - `lpr` commands / 7-9
 - Commands for general use / 7-10
 - Commands for line printer administrators / 7-10
 - Steps to set-up another printer in the `lpr` system / 7-12
 - Troubleshooting the `lpr` system / 7-13
 - `lpr` error messages / 7-13
 - `lpq` error messages / 7-14
 - `lprm` error messages / 7-15
 - `lpd` error messages / 7-16
 - `lpc` error messages / 7-16
 - Writing printer output filters / 7-16
- Setting up a terminal / 7-17
 - The `inittab` file / 7-18
 - Format of `/etc/inittab` / 7-18
 - The `gettydefs` file / 7-19
 - Using another Macintosh computer as a terminal / 7-21
 - Attaching a Macintosh computer as a terminal / 7-21
 - Attaching a VT100, VT102, or other terminal / 7-25
 - Setting up a serial port: `setport` / 7-28

- Setting up a modem / 7-30
 - Dial-out access only / 7-30
 - Dial-in access / 7-32
 - Dialing out on your modem / 7-34
 - Hayes-compatible modems / 7-34

8 Checking the A/UX File System: `fsck` / 8-1

- Introduction to `fsck` / 8-3
- Overview of the A/UX file system / 8-3
 - Blocks and bytes / 8-4
 - Inodes / 8-4
 - Direct and indirect blocks / 8-6
 - More on inodes / 8-7
 - Starting from the top / 8-8
 - Superblock / 8-9
 - Block I/O / 8-10
 - The buffer cache / 8-10
 - Special files and the `/dev` directory / 8-11
 - The contents of device inodes / 8-12
- How `fsck` works / 8-13
 - File system updates / 8-14
 - `fsck` phases / 8-15
 - Phase 1: Check blocks and sizes / 8-15
 - Phase 2: Check pathnames / 8-16
 - Phase 3: Check connectivity / 8-16
 - Phase 4: Check reference counts / 8-16
 - Phase 5: Check cylinder groups (UFS only) / 8-16
 - Phase 5: Check free list (SVFS only) / 8-16
 - Phase 6: Salvage free list (SVFS only) / 8-17
- Using `fsck` / 8-17
 - When to use `fsck` / 8-17
 - `fsck`: a sample interaction / 8-18
 - Multiple file systems and `fsck` / 8-20
- `fsck` error messages / 8-22
 - `fsck` initialization error messages / 8-23
 - Phase 1: Check blocks and sizes / 8-29
 - Phase 1B: Rescan for more duplicates / 8-34
 - Phase 2: Check pathnames / 8-34
 - Phase 3: Check connectivity / 8-44
 - Phase 4: Check reference counts / 8-47

Phase 5: Check cylinder groups (UFS only) / 8-53

Phase 5: Check free list (SVFS only) / 8-54

Phase 6: Salvage free list (SVFS only) / 8-57

Cleanup / 8-57

Restoring missing files / 8-58

Restoring with `cpio` / 8-59

Restoring with `restore` / 8-59

Restoring with `pax` / 8-60

9 System Activity Package / 9-1

The system activity counters / 9-3

The system activity data collector / 9-3

The `sadc` command / 9-4

The `sa1` and `sa2` commands / 9-4

Setting up the system activity functions / 9-5

The system activity reports / 9-6

System activity reports / 9-7

The system activity reporter / 9-7

CPU utilization / 9-9

Buffer activity / 9-10

Swapping and switching activity / 9-11

System call activity / 9-11

Queue activity / 9-12

Message and semaphore activity / 9-13

Device activity / 9-13

Table overflow status / 9-15

File access activity / 9-15

Single-command activity report / 9-17

System activity graphs / 9-18

10 Troubleshooting / 10-1

Trouble while starting up A/UX / 10-3

How A/UX starts up from a hard disk / 10-3

Troubleshooting start up problems / 10-3

Accessing the A/UX file system from A/UX Startup / 10-4

Reinitializing a disk with bad sectors / 10-5

Screen locks, power failures, and emergencies / 10-6

- Troubleshooting user and group administration / 10-7
 - Overriding the default time zone / 10-10
- Errors while changing your password / 10-10
- Other Miscellaneous Issues / 10-11

Appendix A Files Unique to A/UX / A-1

Appendix B Additional Reading / B-1

Appendix C The System V Print Spooler: `lp` / C-1

- `lp` commands / C-3
 - Commands for general use / C-3
 - Commands for `lp` administrators / C-4
- Determining `lp` status / C-5
- The `lp` scheduler / C-5
 - Activating the scheduler / C-6
 - Stopping and starting the `lp` scheduler / C-6
- Configuring the `lp` system / C-8
 - Introducing new destinations / C-8
 - Modifying existing destinations / C-10
 - Altering the system default destinations / C-12
 - Removing destinations / C-13
- Using the `lp` system / C-14
 - Allowing and refusing requests / C-15
 - Allowing and inhibiting printing / C-16
 - Moving requests between destinations / C-17
 - Canceling requests / C-18
- Troubleshooting the `lp` system / C-18
 - Problems starting `lpsched` / C-18
 - Restarting `lpsched` / C-19
 - Repairing a damaged `outputq` file / C-20
- `lp` system files / C-21
 - `lp` system command permissions / C-23

Appendix D System Accounting Package / D-1

Overview / D-3

Starting system accounting / D-3

Updating holidays / D-5

Accounting reports / D-6

Daily summary reports / D-6

Detailed reports / D-12

The `prtacct` procedure / D-14

The `prctmp` procedure / D-15

Routine accounting procedures / D-15

The `startup` procedure / D-16

The `chargefee` procedure / D-16

The `ckpacct` procedure / D-16

The `dodisk` procedure / D-17

The `monacct` procedure / D-17

The `runacct` procedure / D-18

Restarting `runacct` / D-18

In case `runacct` fails / D-19

Error messages / D-21

Fixing corrupted files / D-23

Stopping system accounting / D-25

Appendix E Using the `dp` Utility / E-1

Manipulating slice numbers / E-3

Assigning permanent slice numbers / E-6

Index / IN-1

Figures and Tables

Chapter 2 System Startup and Shutdown

Figure 2-1 Overview of system startup and shutdown / 2-3

Figure 2-2 A/UX Startup command shell window / 2-6

Figure 2-3 Booting dialog box / 2-10

Figure 2-4 General dialog box / 2-11

Figure 2-5 Dialog box displayed by `fsck` / 2-17

Figure 2-6 Changing the A/UX startup device / 2-22

Chapter 3 User and Group Administration

Figure 3-1 Access classes / 3-28

Figure 3-2 Numeric values for `chmod` / 3-31

Figure 3-3 CommandShell representation of permissions of `/users` / 3-36

Figure 3-4 Finder representation of permissions of `/users` / 3-37

Figure 3-5 File-share permissions enabled for `/users` / 3-37

Table 3-1 File type characters / 3-28

Chapter 4 Preparing a Hard Disk for A/UX

Figure 4-1 Two partitions on a disk / 4-4

Figure 4-2 Sample partition map / 4-5

Figure 4-3 Directory hierarchies on separate partitions / 4-7

Figure 4-4 The mounting of file systems / 4-8

Chapter 5 Backing Up Your System

- Figure 5-1 A common backup scheme / 5-4
- Table 5-1 Standard A/UX device files / 5-5
- Table 5-2 Backup media capacities / 5-8
- Table 5-3 Backup utility compatibility / 5-9
- Table 5-4 `cpio` filter requirement / 5-13
- Table 5-5 `tar` blocking and capacity arguments (in blocks) / 5-22
- Table 5-6 `dump.bsd` blocking and capacity arguments / 5-31
- Table 5-7 `dd` blocking arguments / 5-44

Chapter 7 Managing Printers, Terminals and Modems

- Figure 7-1 Signals on a serial port / 7-4

Chapter 8 Checking the A/UX File System: `fsck`

- Figure 8-1 I-number relationships / 8-5
- Figure 8-2 Indirect blocks on a 4K file system / 8-6
- Figure 8-3 File-directory connection through inodes / 8-8
- Figure 8-4 A description of sample entries in `/etc/fstab` / 8-20
- Figure 8-5 How `fsck` decides whether to check a file system / 8-21

Appendix E Using the `dp` Utility

- Table E-1 A/UX partition types available / E-3

About This Guide

A/UX joins two operating systems that have different histories and different styles of operation, but now work together. They are the UNIX[®] operating system and the Macintosh Operating System (Macintosh OS). The version of UNIX that has been adapted to the Macintosh computer is referred to as A/UX when this book discusses topics specific to its operation on the Macintosh; otherwise, the more general name UNIX is used.

Who this manual is for

A/UX Local System Administration is intended for persons who fall into one of the following categories:

- UNIX system administrators who need information on how to administer A/UX.
- A/UX users who are ready to assume more responsibility for managing their systems.

The first group consists of administrators who are integrating A/UX into an existing UNIX environment and need to know specific information that applies to A/UX administration.

The second group consists of advanced users who have mastered A/UX essentials and are eager to explore the next step in understanding and mastering UNIX.

Other documents and manuals that supply additional information, including explanations of concepts that underlie UNIX system operations are available to both groups. Some of these additional resources are listed in Appendix B.

What you need to know

To use the information presented in this manual, you should be familiar with operating the Macintosh under the Finder and with the information contained in *A/UX Installation Guide*, *Setting Up Accounts and Peripherals*, and *A/UX Essentials*. These manuals introduce UNIX and lead you step-by-step through some important A/UX operations. Important concepts described in those manuals include

- installing A/UX
- adding user accounts
- adding peripheral devices
- partitions and file systems
- file access permissions
- login accounts
- pathnames, files, folders, and directories
- using the CommandShell
- using TextEditor
- customizing your work environment

Some of the same administrative tasks are described in both *A/UX Essentials* and *A/UX Local System Administration*. Where this is true, *A/UX Essentials* provides the procedures using Commando or the Finder interface. *A/UX Local System Administration* provides the procedures using the command shell interface, and more administrative information.

What this manual contains

Here is a description of what is covered in *A/UX Local System Administration*:

- Chapter 1, “Concepts of System Administration,” introduces system administration and provides suggestions to help you get started. Concepts such as administrative logins, single-user and multi-user mode and the two different UNIX file systems supported by A/UX are discussed. Information such as setting the system time and configuring a new kernel are also discussed.

- Chapter 2, “System Startup and Shutdown,” discusses the startup process and the A/UX Startup program, which you can use to modify the A/UX environment from the Macintosh Operating System. Logging out, restarting the system, and shutting down the system are also presented.
- Chapter 3, “User and Group Administration,” presents the procedures to manually add and remove users from the system. In addition, the user’s working environment is discussed along with groups, files, directories, and system security.
- Chapter 4, “Preparing a Hard Disk for A/UX,” expands on the installation procedures provided in *A/UX Installation Guide* and discusses partitions, slices, and commands that create and mount file systems.
- Chapter 5, “Backing Up Your System,” describes backup schemes and commands that you use to back up your system.
- Chapter 6, “Managing Disks,” describes autorecovery procedures and how to manage your disk space effectively. Included in this chapter are recommendations for compressing files, and mounting CD-ROM drives.
- Chapter 7, “Managing Printers, Terminals, and Modems,” focuses primarily on the Berkeley print spooler program, `lpr`, and how to administer the spooler. Recommendations for attaching a terminal and modem to your system are also discussed.
- Chapter 8, “Checking the A/UX File System: `fsck`,” describes in detail the `fsck` program and how it is used on both the Berkeley and System V file systems.
- Chapter 9, “System Activity Package,” describes the programs that monitor and report on system activity.
- Chapter 10, “Troubleshooting,” describes common problems and error messages encountered during system operation and offers appropriate response actions. In addition, it provides the procedures to access the A/UX file systems with A/UX Startup utilities.
- Appendix A, “Files Unique to A/UX,” contains a list of commands that are only found in A/UX.
- Appendix B, “Additional Reading,” lists a selection of reference works about the UNIX operating system.
- Appendix C, “The System V Print Spooler: `lp`,” describes the System V print spooler program, `lp`.

- Appendix D, “System Accounting Package,” describes the routine accounting procedures that run on an A/UX system.
- Appendix E, “Using the `dp` Utility,” describes the use of a UNIX partitioning utility, whose function has been largely replaced by the Hard Disk SC Setup application.

Conventions used in this guide

A/UX guides follow specific conventions. For example, words that require special emphasis appear in specific fonts or font styles. The following sections describe the conventions used in all A/UX guides.

Keys and key combinations

Certain keys on the keyboard have special names. These modifier and character keys, often used in combination with other keys, perform various functions. In this guide, the names of these keys are in Initial Capital letters followed by SMALL CAPITAL letters.

The key names are

CAPS LOCK	DOWN ARROW (↓)	OPTION	SPACE BAR
COMMAND (⌘)	ENTER	RETURN	TAB
CONTROL	ESCAPE	RIGHT ARROW (→)	UP ARROW (↑)
DELETE	LEFT ARROW (←)	SHIFT	

Sometimes you will see two or more names joined by hyphens. The hyphens indicate that you use two or more keys together to perform a specific function. For example,

Press `COMMAND-K`

means “Hold down the `COMMAND` key and press the `K` key.”

Terminology

In A/UX guides, a certain term can represent a specific set of actions. For example, the word *enter* indicates that you type a series of characters on the command line and press the RETURN key. The instruction

Enter `ls`

means “Type `ls` and press the RETURN key.”

Here is a list of common terms and the corresponding actions you take.

<i>Term</i>	<i>Action</i>
Click	Press and then immediately release the mouse button.
Drag	Position the mouse pointer, press and hold down the mouse button while moving the mouse, and then release the mouse button.
Choose	Activate a command in a menu. To choose a command from a pull-down menu, click once on the menu title and, while holding down the mouse button, drag down until the command is highlighted. Then release the mouse button.
Select	Highlight a selectable object by positioning the mouse pointer on the object and clicking.
Type	Type an entry <i>without</i> pressing the RETURN key.
Enter	Type the series of characters indicated and press the RETURN key.

The Courier font

Throughout A/UX guides, words that you see on the screen or that you must type exactly as shown are in the Courier font.

For example, suppose you see this instruction:

Type `date` on the command line and press RETURN.

The word `date` is in the Courier font to indicate that you must type it.

Suppose you then read this explanation:

Once you press RETURN, you'll see something like this:

```
Tues Feb 11 17:04:00 PDT 1992
```

In this case, `Courier` is used to represent exactly what appears on the screen.

All A/UX reference manual page names are also shown in the `Courier` font. For example, the entry `ls(1)` indicates that `ls` is the name of a manual page in an A/UX reference manual. See the section “Manual Page Reference Notation” later in this preface for more information on A/UX command reference manuals.

Font styles

Italics are used to indicate that a word or set of words is a placeholder for part of a command. For example,

```
cat filename
```

tells you that *filename* is a placeholder for the name of a file you wish to view. If you want to view the contents of a file named `Elvis`, type the word `Elvis` in place of *filename*. In other words, enter

```
cat Elvis
```

New terms appear in **boldface** where they are defined. Boldface is also used for steps in a series of instructions, and to identify error messages.

A/UX command syntax

A/UX commands follow a specific command syntax. A typical A/UX command gives the command name first, followed by options and arguments. For example, here is the syntax for the `wc` command:

```
wc [-l] [-w] [name...]
```

In this example, `wc` is the command, `-l` and `-w` are options, *name* is an argument, and the ellipses (...) indicate that more than one argument can be used. Note that each command element is separated by a space.

The following list gives more information about the elements of an A/UX command.

<i>Element</i>	<i>Description</i>
<i>command</i>	The command name.
<i>option</i>	A character or group of characters that modifies the command. Most options have the form <i>-option</i> , where <i>option</i> is a letter representing an option. Most commands have one or more options.
<i>argument</i>	A modification or specification of a command, usually a filename or symbols representing one or more filenames.
[]	Brackets used to enclose an optional item—that is, an item that is not essential for execution of the command.
...	Ellipses used to indicate that more than one argument may be entered.

For example, the `wc` command is used to count lines, words, and characters in a file. Thus, you can enter

```
wc -w Priscilla
```

In this command line, `-w` is the option that instructs the command to count all of the words in the file, and the argument `Priscilla` is the file to be searched.

Manual page reference notation

A/UX Command Reference, *A/UX Programmer's Reference*, *A/UX System Administrator's Reference*, *X11 Command Reference for A/UX*, and *X11 Programmer's Reference for A/UX* contain descriptions of commands, subroutines, and other related information. Such descriptions are known as *manual pages* (often shortened to *man pages*). Manual pages are organized within these references by section numbers. The standard A/UX cross-reference notation is

command (*section*)

where *command* is the name of the command, file, or other facility; *section* is the number of the section in which the item resides.

- Items followed by section numbers (1M) and (8) are described in *A/UX System Administrator's Reference*.
- Items followed by section numbers (1) and (6) are described in *A/UX Command Reference*.
- Items followed by section numbers (2), (3), (4), (5), and (7) are described in *A/UX Programmer's Reference*.

- Items followed by section number (1X) are described in *X11 Command Reference for A/UX*.
- Items followed by section numbers (3X) and (3Xt) are described in *X11 Programmer's Reference for A/UX*.

For example

```
cat (1)
```

refers to the command `cat`, which is described in Section 1 of *A/UX Command Reference*.

You can display manual pages on the screen by using the `man` command. For example, enter the command

```
man cat
```

to display the manual page for the `cat` command, including its description, syntax, options, and other pertinent information. To exit, press the SPACE BAR until you see a command prompt, or type `q` at any time to return immediately to your command prompt.

For more information

To find out where you need to go for more information about how to use A/UX, see *Road Map to A/UX*. This guide contains descriptions of each A/UX guide and ordering information for all the guides in the A/UX documentation suite.



1 Concepts of System Administration

Some suggestions before you begin / 1-3

Customizing your system / 1-5

Administrative logins and groups / 1-7

Single-user and multi-user states / 1-9

Initial processes: `/etc/inittab` / 1-11

Configuring the kernel / 1-13

Two types of UNIX file systems / 1-17

About A/UX files / 1-19



This chapter introduces some concepts of UNIX® system administration for A/UX. The system administrator is the person responsible for maintaining and modifying the way in which the system operates. Usually only one person is designated as the system administrator and given access to the root account. Most maintenance is in the form of regular tasks, such as performing backups, checking the integrity of the file system, or adding new users. Other tasks are done rarely, such as adding a new hard disk drive or upgrading the operating system.

System administration in A/UX is much the same as in other UNIX systems, although A/UX has simplified many of the more common system administration procedures. Some of the simplified procedures are

- system startup and shutdown
- configuration
- adding new devices
- adding users
- maintaining file system integrity

Some suggestions before you begin

If you are an inexperienced system administrator, here are some suggestions:

- Give the root account a password as soon as you first start the system. Log in as root only when absolutely necessary. Keep the root password secret. For information on the root account see “Administrative Logins” later in this chapter.
- When you use the system as root, you have special privileges and can make changes that affect system operation, such as modifying files that the operating system depends on. You also have the power to make changes that might damage your system. Therefore, when you are changing a file as root, be extra careful to check your work. (When you are logged in as a normal user rather than as root, you are protected from making changes that affect system operation.)
- Make copies of system files before you change them. One way is to copy the original, say it is called `filename`, to `filename.orig`. Using the same base name keeps both versions of the files together in an `ls` output, while putting an extension on the file helps you remember the reason for the changes. Once you have a copy of a file handy, if you make a change to the file that doesn't do what you thought it would, you can easily copy the known working version back over to the original name.
- Back up your hard disks. Backups are your insurance against losing data if something goes wrong. How often you perform backups depends on how much activity there is on your system and how much work you're willing to lose. See Chapter 5, “Backing Up Your System.”
- Maintain a hard-copy system administrator's log notebook. In this log write down what you do and why you do it, as well as when you do it and its effect on the system. A log is important for training—you save time when you can look up answers—and for diagnosing and troubleshooting your system. Often a problem can be solved more easily if you have a clear idea of what changes had been made to the system just prior to the occurrence of the symptom.
- Use `cron` to automate system administration duties or to remind yourself to do them. See the section on `cron` in Chapter 5, “Backing Up Your System,” for instructions.

- For certain system files it is handy to have hard-copy records of the contents of the files. Print a hard-copy version of the files `/etc/passwd`, `/etc/group`, `/etc/hosts`, `/etc/inittab`, and `/etc/fstab`, and store them in a safe place. If you then accidentally delete one of these files, you can rebuild the basic essentials of it from A/UX Startup. This allows you to log in and retrieve a backup version. Since the method for rebuilding a file in A/UX Startup does not allow for full-screen editors, learn the basics of the `ed` editor.
- As an administrator, one of your first tasks will be to establish other user login accounts for persons who will need access to the system. You can name these login accounts anything you wish, within some very general guidelines. See Chapter 3, “User and Group Administration.” Login accounts typically have names associated with the people who use them, such as `bobw`, or `susan`.
- Understand the `setuid` and `setgid` permission bits. Some programs can be run as if they were being run with root permissions, and are therefore capable of becoming a security risk. As a system administrator, you should always have a clear understanding of who is capable of logging in to your system and what they do.
- Use the A/UX `find` command to locate unused files and directories. Unused files are often not needed and waste disk space. After using `find`, you can archive these files and then remove them or truncate them. Use the `df` command periodically to monitor the available space on your disk.
- Watch for files that grow automatically (in general, any file containing the letters `log` or `LOG` as part of its name). The system appends information to these files, and they grow continually while the system is running. Regularly delete or truncate these files after backing them up. See Chapter 6, “Managing Disks.”
- Anytime you are faced with a system problem, such as possible loss of data, always plan and think before you act. Many times simple problems are compounded into disasters because the administrator acts before thinking. Here is a list you might keep in mind before attempting to solve a problem:
 1. Analyze what happened.
 2. Plan your action before actually doing it.
 3. Anticipate the consequences of implementing your plan.
 4. Act.
 5. Record what happened, what you did, and how the system responded.

- Consider attending a UNIX system administration class to learn more about UNIX if you are an inexperienced administrator. Also, subscribe to UNIX-oriented journals and join UNIX-oriented associations to help you stay aware of UNIX trends. See Appendix B for a list of resources.

The remainder of this chapter describes miscellaneous system administration issues you need to be aware of.

Customizing your system

In order to distinguish your system from others in your work group or on your network, it is customary to give it a name. You may also customize the message that is printed out as each user logs in and the time zone in which their system is located.

Naming the system

The default name that A/UX supplies to your system is `localhost`. You may rename the system by choosing another name. This name may be as long as 14 characters and must begin with a letter; it can include uppercase and lowercase letters of the alphabet as well as numbers and underscores. There are two ways you can change this name. If you are putting the system on a network, when you configure the system you will be prompted for a system name, or host name, as part of that process. Otherwise you may give the system a name by creating or editing the file `/etc/HOSTNAME`. This file should contain two fields, separated by spaces or a tab. The name of your system is the word in the first field of this file, and the name of your domain is the second field. (For information on what a domain is, see *A/UX Networking System Administration*). Change the first field and save the file. For example, to name the system `picasso`, the `/etc/HOSTNAME` file might look like the following

```
picasso    none
```

This change takes effect the next time you reboot, or restart, the system. For more information, see `HOSTNAME(4)`.

Changing the message of the day

A file called `/etc/motd` (for “message of the day”) can be used to send messages to all users that log in to the system. You can change the contents of `/etc/motd` to anything you like by editing the file with a text editor. When a user logs in using one of the standard shells, the message is sent to their screen. The message also appears in the first CommandShell window opened in each login session.

Whenever you change the message, the new message will be displayed the next time anyone logs in.

Setting the system time

Both the A/UX and Macintosh operating systems normally keep track of the correct time without intervention. Set the time for A/UX when you first set up your system, as described in *A/UX Installation Guide*, and whenever

- the time zone changes
- the time is incorrect

The two operating systems maintain the time differently but share a single hardware clock. A/UX stores the local time as an offset from Greenwich Mean Time (the GMT bias) and can adjust automatically for daylight saving time. When A/UX is running during the changes of daylight savings time, it automatically correctly synchronizes the Macintosh Operating System clock.

However, if you are running the Macintosh Operating System (Macintosh OS) when daylight saving time changes, you should set the system clock to the correct time before booting A/UX.

Important The only time you should change the Macintosh OS time is if the Macintosh OS is running when daylight savings time changes.

The system should always have the correct time. You can set the time in A/UX (with the `date` command) and have it take effect for both the A/UX and Macintosh operating systems. If you are logged in as `root`, you can change the system time through the General Controls Control Panel. A/UX uses the time and date to monitor and control a number of activities. Without the correct time, the A/UX system cannot properly perform the following tasks:

- log system events, such as mail activity and logins
- record file activity, such as the creation, modification, or accessing of a file
- track file status with utilities such as `make`
- schedule system and user tasks, such as removing core files and making file backups
- run NFS

Resetting after moving a system to a different time zone

If you move the computer to a different time zone, adjust the GMT bias using the `settimezone` command. See the *A/UX Installation Guide* and `settimezone(8)` for more details.

Administrative logins and groups

As described in *A/UX Essentials*, when first installed, A/UX contains two user login accounts. The first is `start`. This login account provides a starting point for users who are new to A/UX and are using the introductory tutorials.

A/UX also provides a Guest account, which lets a guest user have quick access to the system to run Macintosh programs. Since both the `start` and Guest accounts allow easy access to the system, you may wish to either remove them or give them a password to provide better system security. For more information on system security, see “Security” in Chapter 3.

Administrative logins

Administrative logins are those login accounts used only by system administrators or programs that perform specialized system tasks. The `root` login is the only administrative account you will use; other administrative logins are used by programs. If you accidentally remove any of the administrative logins, the system may operate strangely or not at all. The default administrative logins, found in the `/etc/passwd` file, are as follows:

<code>root</code>	The user on the system that can change any permissions and perform any actions. As shipped, the home directory is <code>/</code> , and the shell is <code>/bin/sh</code> . Avoid unusual shells and home directories not on the root volume.
<code>daemon</code>	The owner of certain noninteractive background processes that handle persistent system services, such as network communication and the print spooler.
<code>bin</code>	The owner of most normal commands and system directories in which those commands are stored.
<code>sys</code>	The owner of certain system files, such as <code>/etc/zoneinfo</code> and files used by <code>autorecovery</code> .
<code>adm</code>	The owner of most system accounting programs and directories, in particular <code>/usr/adm</code> . See Appendix D, “System Accounting Package.”
<code>uucp</code>	The owner of programs and directories associated with the UUCP communications package.
<code>nuucp</code>	The login optionally assigned to incoming <code>uucp</code> requests. See <i>A/UX Network System Administration</i> for more information.
<code>lp</code>	The owner of commands and processes associated with the <code>lp</code> line printer spooling and printing package. See Appendix C, “The System V Print Spooler: <code>lp</code> ,” for more information on the <code>lp</code> system.
<code>nobody</code>	Assigned as the default login for remote <code>root</code> access under Network File System (NFS). See <i>A/UX Network System Administration</i> for further information.

Administrative groups

Similar to administrative logins, **administrative groups** help you perform specialized system tasks. As mentioned above, the only administrative login you want to actually log in as is `root`. However, when performing administrative tasks, you don't always want to have all the permissions of `root`. One way to avoid this is to create a special user login account (for example, `sysadmin`) and make that user a member of an administrative group.

Administrative groups are found in the `/etc/group` file and include `sys`, `bin`, `root`, `daemon`, `adm`, `uucp`, `lp`, `utmp`, and `mail`. If, for example, user `sysadmin` is a member of the group `lp` (for line printer administration), then the system administrator can log in under the `sysadmin` account and run `lp` administrative programs, read and write `lp` group affiliated files, and gain access to directories that have group permissions assigned to `lp`. Since you are only affiliated with `lp` administrative permissions you avoid the risks of making an error with the global permissions of `root`. See Chapter 3, "User and Group Administration," for more information on groups.

Single-user and multi-user states

Unlike the Macintosh operating system, A/UX has different run levels, also called run *states* or run *modes*. The system automatically starts up in what is called multi-user state, the preferred state for most activities.

The two basic run levels—single-user and multi-user—will probably suffice for normal operation. Descriptions of these two modes follow. Other run levels are available, but are generally not used. See `init(1M)` for more information on the available states of your A/UX system.

Single-user state

Single-user state is one of several run levels available on the system. As the name implies, in **single-user state**, only one person has access to the computer. This access is available only from the system console. In single-user mode, there is not as much background activity, such as the availability of printers, terminals, and modems on the system. This quiet state of the machine is necessary for the performance of administrative tasks, such as

- Checking the integrity of the system with `fsck` (although it is better to check it from A/UX Startup). For information on `fsck`, see Chapter 8.
- Copying user files to backup media

If you want a Finder interface available while in single-user mode, enter on the command line

```
/mac/bin/mac32
```

Otherwise, the system console command line interface is your window. Also, the person with availability to the system console during single-user mode has all the privileges of `root`. You can create or destroy anything on the system—files, directories, or programs—with just a few keystrokes. This level of power is necessary to perform administrative tasks, but you need to use it selectively. Unless you need the advantages of the quiet state of the system, run A/UX in the multi-user state. This strategy decreases the potential of making a mistake that could damage the system. Also, since single-user mode is not protected with an account login program, you don't expose the system to other users who should not have the privileges of `root`.

To start A/UX in single-user mode, get to the A/UX Startup command line, described in the next chapter. Enter the command

```
launch -S
```

To go from multi-user mode to single-user mode, use the command line form of the `shutdown` command, also described in Chapter 2.

To start A/UX in single-user mode automatically each time the system is started, you need to edit the file `/etc/inittab`. The only reason you would want to do this is on a temporary basis to troubleshoot a specific problem with the system. Change the line

```
is:2:initdefault:
```

to

```
is:s:initdefault:
```

Multi-user state

The default and preferred run level for A/UX is multi-user mode. This run level is recommended even if you are the only user on a machine because it guards against inadvertent system damage. In **multi-user mode**, most system programs and activities are available.

If you double-click the A/UX Startup icon and you have not specifically set A/UX Startup to remain in single-user mode when starting A/UX, you will enter multi-user mode and see the Login dialog box.

To enter multi-user mode from single-user mode, enter on the command line

```
init 2
```

You should see

```
INIT: New run level: 2
```

As the system enters into multi-user mode, commands that are part of the startup process are executed to set up the system. Some of these commands are stored in the files `/etc/inittab`, `/etc/rc`, `/etc/brc`, and `/etc/bcheckrc`. (see “Initial Processes: `/etc/inittab`,” later in this chapter). If this process is successful, you will see the Login dialog box. The system is now running in multi-user mode, and you may log in. If you are not sure whether you are in single-user or multi-user state, enter `exit` on the command line. If you get a login prompt or Login screen, you are in multi-user state.

Initial processes: `/etc/inittab`

A/UX, and all UNIX systems, allocates system resources to users and programs through the use of **processes**. The process that the system creates upon startup, which has a process identification number of 1, is called `init`. (For more information on viewing processes see `ps(1)`. For more information on how processes are used to run the system, refer to entries in Appendix B.) The following is an overview of some the processes executed as the system starts. For more complete information, refer to the `init(1M)` manual page.

The `init` process spawns, or creates, child processes. These child processes help `init` manage all the activities of the system.

One of the first activities `init` does is to check the file `/etc/inittab` for other tasks to perform. In A/UX, the next task is to execute the `/etc/sysinitrc` shell program, which performs basic functions such as setting the system's time zone.

Next, `inittab` specifies that `init` bring the system to the default initial run level. This is the level to which `init` will take A/UX unless interruptions are made while the system is starting. The line in `inittab` that shows us the default run level is

```
is:2:initdefault:      #First Init State
```

The format of the `inittab` file is discussed in Chapter 7. For the purpose of this example, note that the `2` in this line denotes run level 2, which is multi-user mode. Once the initial run level is determined, `init` processes only those `inittab` entries whose *run-level* field is the same as the run level currently in effect.

If you view the contents of `inittab` on your system, you will see the following processes that are invoked by `init`. (In the file, the process names are preceded by an *id* and a *run-level*, as discussed in Chapter 7.)

<code>/etc/bcheckrc</code>	A startup script that runs <code>fsck</code> on those file systems other than <code>root</code> . See Chapter 8 for information on <code>fsck</code> .
<code>/etc/brc</code>	A startup script that sets the permissions on pseudo-ttys.
<code>/etc/rc</code>	A startup script that mounts the file systems (if applicable) and performs other general housekeeping. For a discussion of file systems, see Chapter 8.
<code>/etc/getty</code>	A process that enables logins on serial ports and the console. For a discussion of <code>/etc/getty</code> , see <code>getty(1M)</code> and Chapter 7.

To understand more fully what each of these startup scripts accomplishes, you may want to look at them to understand where they fit in the A/UX Startup process (described in the section “The Startup Sequence” in Chapter 2). This will aid your understanding of the system, and allow you to better pin down a point of failure on startup, should one ever occur.

Changing run levels: `init`

In addition to the `init` process, which must be active the entire time the system is running, it is possible to change run states by invoking `init` on the command line.

When running `init` from the command line, it is possible to disrupt the activities of users currently logged in. For instance, running the `init` command could disrupt an operating modem if you change the entry in `/etc/inittab` that manages the port the modem is connected to. Before making any changes to `/etc/inittab`, know your run states and active processes and make sure that changing them will not disrupt user activity. See `init(1M)`.

Once the system is started and running at the default run level, you can change the run level when logged in as root; enter

```
init run-level
```

on the command line, where *run-level* is an argument to `init` that may be either a value from 0 to 6, or the letters `s` or `S`. (See `init(1M)` in *A/UX System Administrator's Reference*.)

When you enter this command, `init` scans `/etc/inittab`, kills processes that should not be active in the new run level except for those that spawn daemons (see the discussion of `init 0` later in this section), and activates those entries whose *run-level* value is the same as that of the new run level, leaving all other processes untouched.

If you make a change in the `inittab` file that you want the system to recognize, but do not want to change the run level of the system, enter

```
init 0
```

This command forces `init` to reread `/etc/inittab`. If you want to stop a process that is not controlled by an entry in `/etc/inittab`, use the `kill` command.

Configuring the kernel

Each UNIX system has a special file called a kernel. The **kernel** is made up of a number of parts, or modules, each of which is responsible for managing a piece of the system. Some of these modules provide process scheduling while others control aspects of the system, such as disk drives or memory. The kernel is called `/unix` on A/UX and other System V derivative systems.

It is not possible to view the kernel, which is a binary file, in the same way that you can view a text file, but you may obtain a list of the modules that are in the kernel with the command

```
module_dump /unix
```

Through the A/UX kernel, the system associates the software controlling a device with the hardware installed in each NuBus™ slot in the Macintosh computer. When you remove hardware from a system, or change the slot position of a controller board, on the next boot a new kernel is automatically built, with the `autoconfig` program, to reflect the change.

When you add controller boards to the system, the system needs to be reconfigured to recognize the addition so that the hardware may be accessed. This is not done automatically. You must build a new kernel which incorporates the necessary modules with the `newconfig` command.

Autoconfiguration

The `autoconfig` program runs automatically while A/UX is starting. Under some circumstances, `autoconfig` builds a new kernel. The circumstances are:

- Hardware is removed from the system, for example, by removing an Ethernet card from your system
- Slot positions are changed in your system, for example, by moving the Ethernet card from the first slot to the second slot

Once `autoconfig` rebuilds a new kernel it presents a dialog box that states that the system needs to be restarted; this means that you should click the Restart button.

◆ **Note** If `autoconfig` notices the presence of a card for which no module is configured in the kernel, it displays a warning message if you start up with a `launch -v` ◆

To turn off autoconfiguration on startup, change the AutoLaunch command in the Booting dialog box of A/UX Startup to

```
launch -n
```

For more information on this dialog box, see Chapter 2.

The `newconfig` program

The `newconfig` program provides an easy way to build a new kernel to include new features or hardware in the kernel. If new hardware is added to the system and there is no module in the kernel for that hardware, the system can not use the hardware until you run `newconfig`.

The `newconfig` program runs both the `newunix(1M)` and the `autoconfig(1M)` programs automatically. The `newconfig` program takes predefined software modules and either installs them into or removes them from a new kernel. The new kernel is automatically installed into `/unix`. For a complete list of the available software modules and capabilities of `newconfig`, see `newconfig(1M)`. Note that other software manufacturers may add to the modules that `newconfig` can incorporate into the kernel.

◆ **Note** You do not need to use `newconfig` to add terminals, printers, modems, most tape drives, CD-ROM drives, hard disks, or Apple floppy disk drives to the A/UX system. Support for these devices is already built into the standard A/UX kernel. ◆

Follow these steps to run the `newconfig` program.

1 **Check which modules are currently configured in the kernel.**

To do this, enter the command

```
module_dump /unix
```

Note that `newconfig` does not prevent you from making a kernel that does not work. Since the new kernel does not take effect until you restart the system, you may want to use the `module_dump` command to check the contents of the kernel both before and after running the `newconfig` command.

Usually when a kernel does not work, it is because there are dependencies between modules that are not satisfied. For instance, the `nfs` and `cslip` modules both rely on `bnet`; likewise `adsp` relies on `appletalk`. It is a good idea to use the `-v` option to `newconfig` to be able to view any errors you might otherwise not see. Also, *always* choose Restart from the Special menu *immediately* after the `newconfig` command completes building a new kernel.

2 **Run the `newconfig` command.**

To add a new software module or driver to the kernel, specify the device driver or software module you want to add to the kernel as a parameter to `/etc/newconfig`. For example, the command

```
/etc/newconfig -v cslip
```

installs the module for the Compressed Serial Line Interface Protocol (CSLIP). More than one module can be installed at a time; for example

```
/etc/newconfig -v nfs debugger
```

installs the Network File System and A/UX kernel debugger support to the kernel.

◆ **Note** Before using `newconfig` to make a new kernel, have any hardware installation or modifications completed. ◆

To remove a software module from the kernel, specify the module that you want to remove as a parameter to `newconfig` by putting the word `no` before the name of the module. Note that there is not a space between the word `no` and the module. For example,

```
/etc/newconfig -v nodebugger
```

removes the `debugger` module from the kernel.

3 **Restart the system to use the new kernel.**

Follow the directions given in Chapter 2, “System Startup and Shutdown,” to restart the system before using the newly added device or the new software module. Problems could occur if you continue to run the system after you have changed the kernel but have not restarted.

Changing kernel parameters

Under normal conditions, kernel parameters will not need to be changed. Sometimes, however, you may exceed the kernel-configured limits on your system. Such limits are evident from error messages that appear. Some of these messages are:

<i>Error message</i>	<i>Parameter requiring an increase</i>
<code>inode: table is full</code>	<code>NINODE</code>
<code>file: file table is full</code>	<code>NFILE</code>
<code>proc: table is full</code>	<code>NPROC</code>

If you encounter one of these messages, use the `kconfig(1M)` command to increase one of these kernel parameters. When increasing these parameters it is best to increase the value by 50 or 100 and then try running the system for a while. If the error message still occurs, increase the value a little more, and then test again. It is possible to decrease the performance of the system if you increase these values too much.

◆ **Note** The `newconfig` command will alter these parameters when it is building a kernel, based on the type of kernel it is building. Under normal circumstances, you will not need to change these parameters. ◆

You must restart to put the new parameters into effect because `kconfig` changes the kernel file on disk and not the currently running kernel.

Two types of UNIX file systems

Over the years, two major variants of the UNIX operating system have evolved from the original operating system developed by Bell Laboratories. One variation of UNIX has evolved at University of California at Berkeley; the other variation of UNIX has evolved at AT&T. Both versions are similar in that they use many of the same commands (sometimes with different options). However, the way in which the operating system itself accesses

files and file systems differs. What is important is not the differences but rather that you are aware that two different file system structures are available and that A/UX can support either variation. For more information on these different file systems, acquire some of the references in Appendix B, “Additional Resources.”

The Berkeley file system is usually referred to in this manual as **UFS** (UNIX File System), but in some cases as **4.2** (a standard version number) file system.

The AT&T UNIX file system, also known as the System V file system, is usually referred to in this manual as **SVFS** (System V File System), but in some cases as the **5.2** (a standard version number) file system.

You might ask, why two file systems? UFS is the *default* file system type for the root file system when A/UX 3.0 is installed. It has the advantages of faster performance and longer file and directory names, and a simpler method of creating additional file systems.

By contrast, SVFS is provided primarily for compatibility with previous releases of A/UX. Using the SVFS file system can have its advantages when A/UX systems are used in environments where other UNIX systems employ SVFS and their administrators are more familiar with administering SVFS-based systems.

The following file system utilities are common to both UFS and SVFS: `fsck`, `fsdb`, `mount`, and `umount`.

Some commands must read the file system structure in order to work, which is why A/UX provides two versions of these commands. To view some of the commands provided for the two different types of file systems, use the `ls` command on the `/etc/fs/ufs` and `/etc/fs/svfs` directories. You will see commands such as:

<code>clri</code>	<code>fsdb</code>	<code>mkfs</code>
<code>df</code>	<code>fsirand</code>	<code>ncheck</code>
<code>dump.bsd</code>	<code>fsstat</code>	<code>rdump</code>
<code>fsck</code>	<code>fstyp</code>	<code>restore</code>

- ◆ **Note** Do not confuse the Macintosh hierarchical file system (HFS) with a UNIX file system. HFS can only be accessed through the Finder, and is very different from UFS or SVFS file system. ◆

About A/UX files

A list of the files that make up the A/UX release can be found in the file `/FILES`, which gives the full pathname of A/UX system files along with a short description. The list is useful when you want to quickly find a short description of a system file. This file is sometimes used by the `autorecovery` program and should remain on the system.

If you have not installed all the packages from the A/UX release, there will be files listed in `/FILES` that are not present on your system




2 System Startup and Shutdown

Starting up the system / 2-4

About A/UX Startup / 2-5

Logging out, restarting, and shutting down / 2-18

Changing the A/UX startup device / 2-22



This chapter provides an overview of what A/UX does upon startup and shutdown. Understanding the way A/UX starts up involves understanding the different facets of the application called A/UX Startup. A/UX Startup, if uninterrupted, will load the A/UX kernel into memory and transfer control of the system to the kernel. If you interrupt A/UX Startup (before control is transferred), you are presented with an A/UX Startup command shell. This command shell provides a set of utilities to troubleshoot and manipulate A/UX while the computer is still governed by the Macintosh OS. Once you are running A/UX, there are different ways to shut down the system. Shutdown procedures, once initiated, run automatically and cannot be interrupted.

▲ **Warning** Always follow the system shutdown steps described in this chapter; otherwise you risk losing or damaging data stored on your hard disks. ▲

◆ **Note** If you *always* want to work with A/UX, you can set your system such that A/UX will automatically launch whenever you start the system. For instructions, see *A/UX Essentials*. ◆

Refer to Figure 2-1 during the following discussion of A/UX startup and shutdown.

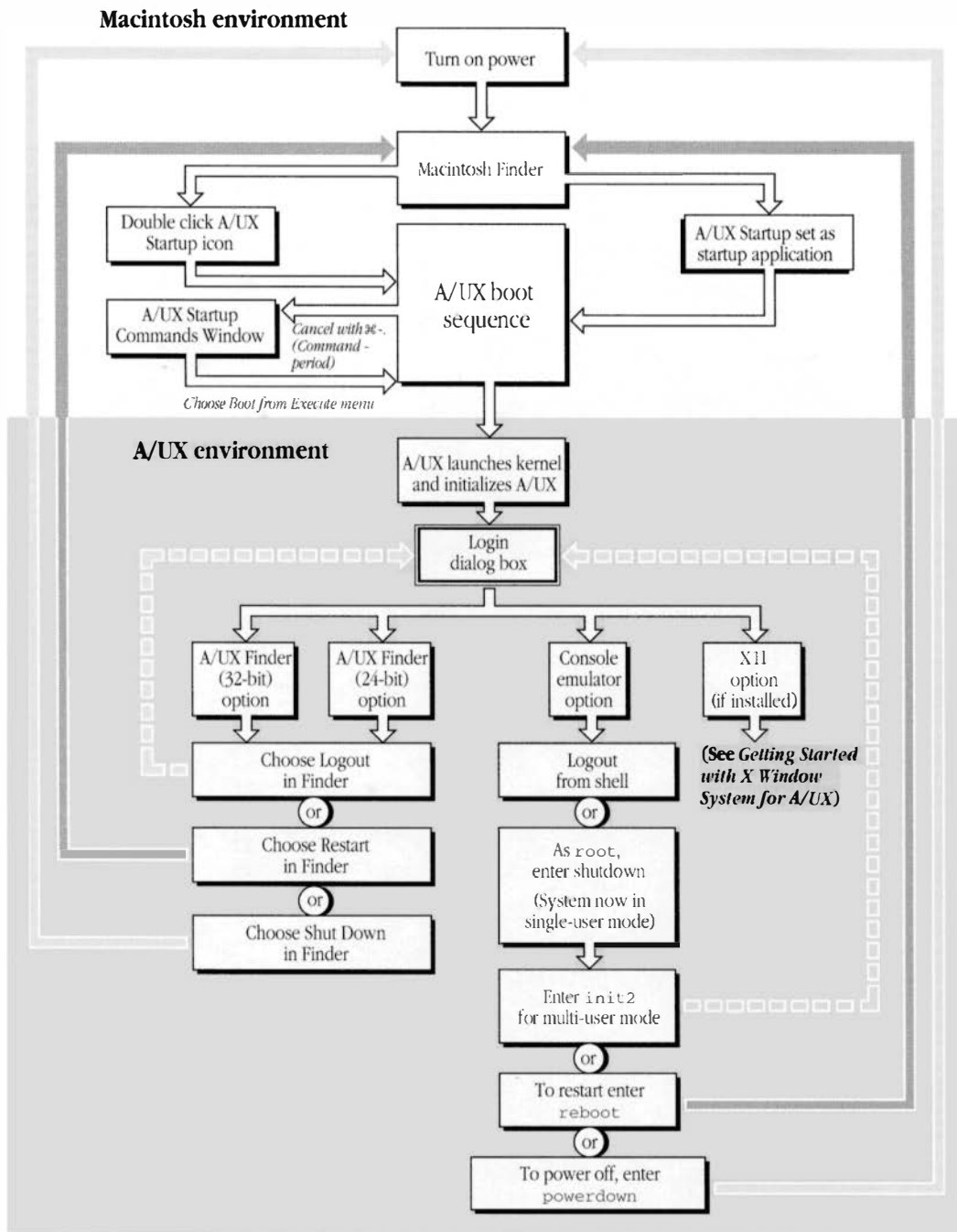


Figure 2-1 Overview of system startup and shutdown

Starting up the system

This chapter assumes you read the *A/UX Installation Guide* and set up your Macintosh computer. It further assumes that you are familiar with the A/UX basics presented in *A/UX Essentials*. This chapter provides more details for a system administrator about procedures documented in other manuals.

Unless you configure it otherwise, your system will start up in the Macintosh Finder by default. To boot A/UX, double-click the A/UX Startup icon. Figure 2-1, "Overview of System Startup and Shutdown," shows the division of the startup procedure between the Macintosh environment and the A/UX environment. A/UX Startup, by default, is on the disk named MacPartition.

◆ **Note** You can not return to the Macintosh OS from A/UX without restarting the system. ◆

A/UX is started like an application from the Macintosh OS, similar to other applications such as MacWrite® or MacPaint®. However, unlike other applications, once started it takes over control of the system from the Macintosh OS. After A/UX finishes starting up, you have both UNIX and Macintosh System 7 features available to you. At this point you can run both compatible Macintosh applications and UNIX commands side by side.

If you have installed A/UX on your disk without customization, you will have a small disk partition called MacPartition. This partition contains a System Folder, a folder named `bin`, and the A/UX Startup application. The System Folder has some very basic features designed to start a limited version of Macintosh System 7 within a minimum amount of disk space. The `bin` folder contains utilities to run from within the A/UX Startup environment to check or repair your A/UX files systems before A/UX is started. A/UX Startup is the Macintosh program that checks your disk, loads A/UX into memory, and starts up A/UX in a similar way to other UNIX operating systems.

By default, and as long as there is nothing wrong with the system, A/UX Startup completes the entire process of loading A/UX with no intervention on your part. However, you can cancel the startup process and use the A/UX Startup utilities. These are available within a window which is similar to the UNIX command line interface. Many of the same commands and usage conventions available in UNIX are also available in A/UX Startup: shell variables, comments, and input and output redirection. For system

administrators who are familiar with UNIX, the command line interface of A/UX Startup is similar to the stand-alone shell capability of other versions of UNIX.

However, for all its similarity to UNIX, A/UX Startup is a Macintosh program that allows you to manipulate the UNIX file system. It runs read-only from a Macintosh file system.

For more information on A/UX Startup's capabilities, see "A/UX Startup Menus" later in this chapter, and `StartupShell(8)`.

◆ **Note** The command line interface of A/UX Startup provides a user with all the privileges of `root`. Therefore, it is a good idea to password protect A/UX Startup by checking the Password checking box in the General dialog box. See "A/UX Startup Menus" later in this chapter for how to password protect A/UX Startup. You don't need to enter this password to start up A/UX; password protection is only enforced for the use of the troubleshooting and repair utilities ◆

About A/UX Startup

This section describes the A/UX Startup menus and options, A/UX Startup command shell window, and UNIX system-like commands.

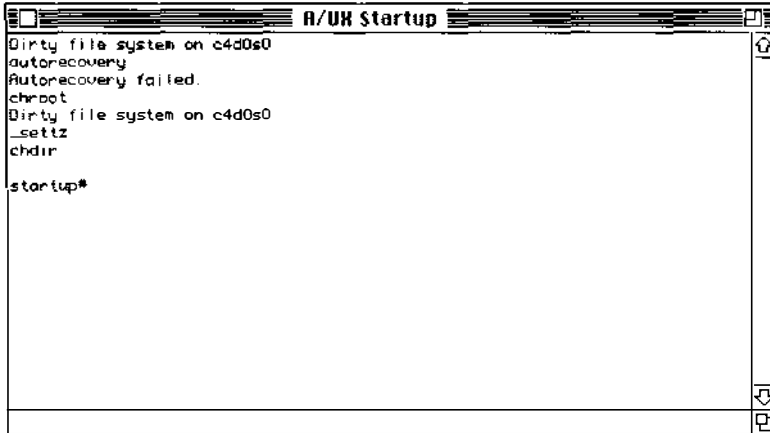
A/UX Startup command shell window

Initially, to enter A/UX Startup you cancel the startup process for A/UX. If you then deselect the Automatically Boot at startup box in the Booting dialog box under the Preferences menu of A/UX Startup, you will enter the A/UX Startup command window each time you double-click the A/UX Startup icon. (See Figure 2-3 for a picture of the Booting dialog box.) Once in the A/UX Startup shell, you see a command line prompt. The default prompt is

```
startup#
```

You can change the prompt to another string by redefining the shell variable `Prompt`. There are other shell variables you are able to customize if you prefer. These are `PATH`, `TZ`, `HOME`, and `ROOT`.

- ^ **Important** When you cancel the startup process for A/UX, you will see the A/UX Startup command shell window. If the top lines of the window denote a dirty file system, as shown in Figure 2-2, you must run the `fsck` command on that file system before changing any files. Failure to run `fsck` may result in corruption of the file system. ^



```
A/UX Startup
Dirty file system on c4d0s0
autorecovery
Autorecovery failed.
chroot
Dirty file system on c4d0s0
settz
chdir
startup*
```

Figure 2-2 A/UX Startup command shell window

Commands that run in A/UX Startup

There are two sets of commands that are available to you in A/UX Startup. One set is in the `bin` folder in MacPartition:

<code>cat</code>	<code>dp</code>	<code>mv</code>	<code>svfs:fsdb</code>
<code>chgrp</code>	<code>ed</code>	<code>newfs</code>	<code>svfs:mkfs</code>
<code>chmod</code>	<code>fsck</code>	<code>od</code>	<code>tar</code>
<code>chown</code>	<code>kconfig</code>	<code>pname</code>	<code>tunefs</code>
<code>cp</code>	<code>ln</code>	<code>rm</code>	<code>ufs:fsck</code>
<code>cpio</code>	<code>ls</code>	<code>settz</code>	<code>ufs:fsdb</code>
<code>date</code>	<code>mkdir</code>	<code>stty</code>	<code>ufs:mkfs</code>
<code>dd</code>	<code>mknod</code>	<code>svfs:fsck</code>	

The second set are also UNIX system-like commands. These commands are built into A/UX Startup. Remember that these are not UNIX commands, but are functionally identical versions of the UNIX commands that you can use to perform UNIX system administration from the Macintosh environment:

auto	echo	pwd	unauto
boot	eject	readonly	unexport
cd	exit	restart	unset
chdir	export	set	
chroot	help	shutdown	
default	powerdown	umask	

◆ **Note** The `boot_cd`, `launch` and `esch` commands are also available in A/UX Startup. These commands are in MacPartition. ◆

Since these commands allow you to work on A/UX files from outside A/UX, they are especially useful for troubleshooting. For example, you might want to edit `/etc/inittab` without first booting A/UX if you suspect that that file is causing you problems during system boot. For a description of this procedure, see Chapter 10, “Troubleshooting.”

A/UX Startup also provides a concise help facility, which is available by typing

```
help
```

Help for individual command is available by typing

```
help name
```

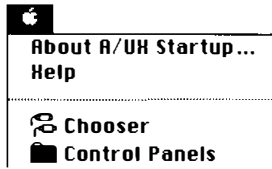
where *name* is the name of one of the commands listed above.

You can also refer to the individual command descriptions in the *A/UX System Administrator's Reference* or *A/UX Command Reference* for more complete information.

A/UX Startup menus

The A/UX Startup menu bar contains the Apple, File, Edit, Execute, and Preferences menus. A brief description of each follows. For additional information, see `StartupShell(8)` in *A/UX System Administrator's Reference*.

Apple menu



About A/UX Startup Displays introductory information about A/UX Startup.

Help Displays the default help messages in the A/UX Startup window plus additional information about the `help` command.

File menu



Close You cannot close the A/UX Startup command shell window, so this item is dimmed.

Quit Exits A/UX Startup and returns you to the Macintosh desktop. This command has the same effect as the `exit` command typed on the command line.

Edit menu

Edit	
Undo	⌘Z

Cut	⌘K
Copy	⌘C
Paste	⌘U
Clear	

Undo, Cut, Copy, Paste, and Clear Only Copy and Paste are enabled in A/UX Startup. Use Copy to copy selected text in the A/UX Startup command shell window, and Paste to place the text last copied to the end of the window.

Execute menu

Execute	
Boot	⌘B
AutoRecovery	
AutoLaunch	⌘L

Kill	⌘K

Restart	
Shut Down	

Boot Performs the AutoRecovery command, and then performs the AutoLaunch command if AutoRecovery was successful. See the Booting dialog box in the Preferences menu for the contents of these commands. This menu item is the same as typing the `boot` command at the A/UX Startup command line.

AutoRecovery Performs the command in the AutoRecovery field of the Booting dialog box in the Preferences menu. Normally this is an `fsck` on the root file system. See Figure 2-8 for the Booting dialog box.

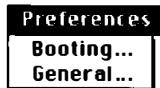
AutoLaunch Performs the command in the AutoLaunch field of the Booting dialog box in the Preferences menu. By default, the command is `launch`.

Kill Stops the currently running program. `COMMAND-PERIOD` and `COMMAND-K` are keyboard shortcuts for this item. Note that when A/UX is running, `COMMAND-K` invokes Commando.

Restart Restarts the computer. It is the same as typing the `restart` command on the A/UX Startup command line.

Shut Down Turns off the computer.

Preferences menu



Booting Provides a dialog box for setting `boot` command startup parameters, including the Boot item in the Execute menu.

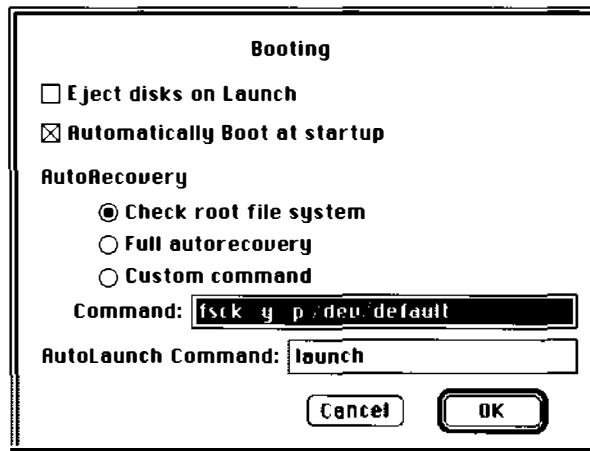


Figure 2-3 Booting dialog box

Fields in this dialog box are described as follows:

- Eject disks on Launch** Select this box to eject all floppy disks when the kernel is launched.
- Automatically Boot at startup** Select this box to run the `boot` command automatically, causing A/UX to start when A/UX Startup is run rather than entering the A/UX Startup command shell. (This is the default setting.)
- AutoRecovery Command** By default, the Check root file system button is selected and the `fsck` command line that is automatically run before the kernel is launched is displayed in the Command box. You can also choose the Full autorecovery button, which changes the command to `esck -b`. Finally, you can select the Custom command button and enter the command of your preference in the Command box. The command in the Command box is run as one of the first steps in the startup process.
- AutoLaunch Command** This text window displays the value of the built-in `autolaunch` variable. Change the value by selecting this box and editing the text. To display console messages during the startup sequence, enter `launch -v` in this field. These messages can be very useful for troubleshooting. See `launch(8)` in *A/UX System Administrator's Reference*.

General Provides a dialog box that contains the following miscellaneous items.

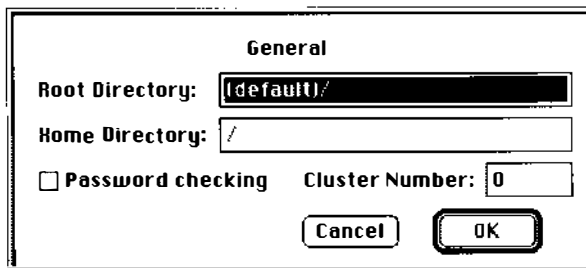


Figure 2-4 General dialog box

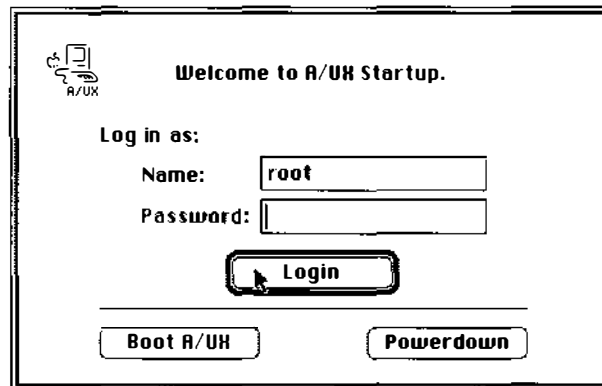
Root Directory This text window displays the value of the built-in `ROOT` variable. To change it, select this box and edit the text. To start A/UX from the same hard disk as A/UX Startup, use `(default) /`. To start A/UX from a different hard disk, use `(ID, 0, 0) /`, where *ID* is the SCSI ID of the hard disk that contains A/UX. When you change the value in this window, the system makes the corresponding change in the Booting dialog box.

Home Directory This field displays the value of the built-in `HOME` variable. To change the value, select this box and edit the text. This variable determines the default directory you are placed in upon entering A/UX Startup.

cluster number This field displays the value of the `autorecovery` cluster number. See `autorecovery(8)` for an explanation of this number.

▲ **Warning** Changing the Cluster Number may cause autorecovery to fail. ▲

Password checking When checked, displays a dialog box requesting a password for a user authorized to use A/UX Startup. This prevents unauthorized users from canceling A/UX Startup and using the A/UX Startup command shell window. You must type a valid user and password (usually the `root` password) to gain access to the A/UX Startup command shell window.



The startup sequence

During the startup, or boot, process, several startup screens appear. Some screens only display when a particular system condition is in effect. When you double-click the A/UX Startup icon the computer automatically starts up to multi-user state unless you designate otherwise, as described in the section, “Single-User and Multi-User States” in Chapter 1.

After you double-click the A/UX Startup icon, the first two windows you see have a Cancel button. When this Cancel button is enabled, you can click it to cancel the startup process and see the A/UX Startup command window. (Another way to cancel the startup sequence is to press `COMMAND-PERIOD (COMMAND-.)` while the Cancel button is showing.) This window allows you to use the utilities provided with A/UX Startup. The remaining windows, display a Messages button. When the Messages button is active, you can click it to view the A/UX System Console window, which shows you information on what is happening during the startup process.

Checking



At this point, the computer is still being run by the Macintosh OS. A/UX Startup first checks the integrity of the root file system, using the `fsck` command, making sure that the file system is undamaged. If `fsck` fails, A/UX Startup has found a problem with the files it needs in order to continue the startup process. See `fsck(1M)` and Chapter 8, “Checking the A/UX File System: `fsck`” for additional information.

By default, A/UX Startup’s `AutoRecovery` option runs `fsck` for the root file system when the root file system mount flag is on, which indicates possible file system damage.

Loading



During this phase, A/UX Startup is loading the A/UX kernel into memory and preparing to hand off the system operation to A/UX. You can still cancel the startup process, return to the Macintosh OS, and operate the A/UX Startup utilities.

- ◆ **Note** The Cancel button operates during the checking and loading phases. ◆

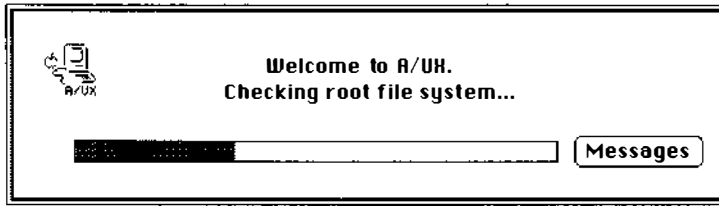
Launching



During launching, A/UX Startup passes control of the system to A/UX and the kernel is initialized. When launching starts, the Cancel button is disabled and is then replaced by the Messages button. The A/UX start up procedure can no longer be stopped. The progress bar does not move during this phase. The screen blinks momentarily, which is part of normal operation.

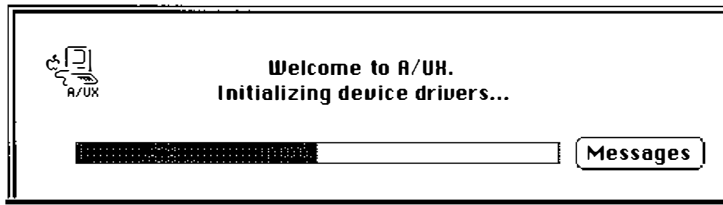
The `/etc/macsysinit` program initializes the Macintosh environment under A/UX for the duration of the startup process.

Checking root file system



The `/etc/sysinitrc` script verifies that the root file system is not damaged; it then mounts the root file system, and executes `autoconfig` and device driver startup scripts. If `autoconfig` detects that the system hardware configuration has changed since the last startup (for example, you have moved or removed a video card), `autoconfig` creates a new kernel. It then prompts you to restart the system. When you start up A/UX again, the new kernel will be used.

Initializing device drivers



During this phase, the device drivers installed in the kernel are initialized. See Chapter 7, “Managing Printers, Terminals, and Modems,” for a discussion of device drivers. Also, if `/etc/HOSTNAME` exists, the host name and domain name of the system are set. If this file does not exist and you have set up a networking kernel, the system prompts you to enter the host name and domain name in the A/UX System Console window. Otherwise the default name is `localhost` and the default domain is `localdomain`. If entering single-user instead of multi-user state, enter it after this step.

Checking file systems

The `/etc/bcheckrc` program runs `fsck`, checking all file systems that appear in the `/etc/fstab` file with a pass number entry of 2 or greater. This phase will only occur if the system needs to mount file systems other than `/` and the file systems that appear in `/etc/fstab`. If this phase occurs, one of two dialog boxes will appear depending on whether the system runs `fsck` on a mount point or on a file system. The dialog box for an example mount point of `/users1` looks like this:



The dialog box for an example device of `/dev/dsk/c3d0s0` looks like this:



The specific mount directories are shown in the dialog box, which allows you to monitor progress and see which file systems are being repaired. If problems are detected for a file system, `fsck` displays a dialog that asks whether or not to proceed with repairs (see Figure 2-5). If you skip the repair process, the startup sequence continues; however, you must run `fsck` on that file system before it can be mounted.

Checking a file system can take longer than any other phase of the startup process, depending on whether the file system has any problems. Remember that a file system must be repaired by using `fsck` before it can be mounted and used.

- ◆ **Note** A common cause of file system damage is failure to follow proper shutdown procedures. Proper shutdown procedures are described in “Logging Out, Restarting, and Shutting Down,” later in this chapter. ◆

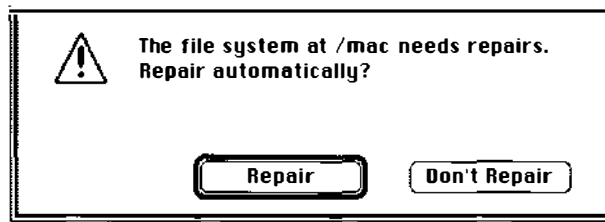


Figure 2-5 Dialog box displayed by `fsck`

Starting background processes



In this final startup phase, general system housekeeping commands contained in the `/etc/brc` and `/etc/rc` scripts execute, and file systems other than root which cleanly passed `fsck` checks are mounted. Next the `init` process creates additional background processes as specified in the `/etc/inittab` file. When the background processes have all started, UNIX initialization is complete and the Login dialog box is displayed. At this point you can log in and enter UNIX commands.

Logging out, restarting, and shutting down

To log out, restart, and shut down the computer from the Finder, use the choices available from the Special menu. For more information, see *A/UX Essentials*. Using the menu items is the recommended procedure. To log out, restart, and shutdown from the command line, follow the instructions below.

- ▲ **Warning** When you finish using A/UX and you want to turn the power off or return to the Macintosh OS, you must use the A/UX `shutdown` command or the Restart or Shut Down menu choices. If you fail to shut down the system gracefully, you risk damaging or losing disk information. To shut down the system, you must be logged in as root or you must know the root password. ▲

Logging out from the command line

When you use the `logout` command, all applications that you opened are closed before the logout action is completed. From a C shell CommandShell window, `logout` only terminates jobs running in that window. To log out of your A/UX account, the recommended procedure is to use the Logout menu item from the Special menu.

Entering single-user state from multi-user state

While the Shut Down command from the Finder turns off the computer, the `shutdown` command from the CommandShell will shut down the computer from multi-user state to single-user state. The `shutdown` command

- stops all daemons and kills all remaining processes
- executes `sync` a number of times to ensure that any data in memory is written to the disks
- unmounts the file systems
- executes `sync` again

◆ **Note** Do not confuse the `reboot` and `powerdown` commands you can perform in A/UX from the CommandShell with the Restart and Shut Down menu items in the Finder. The Restart and Shut Down menu items perform proper system shutdowns from multi-user mode. The `reboot` and `powerdown` commands perform proper system shutdown from single-user mode. ◆

The `shutdown` command also allows you to broadcast a message to all users who are logged in so they will know that the computer is being shut down. Additionally, you can specify a time delay between when the `shutdown` command is issued and when the system power actually turns off.

Using the `shutdown` command

Follow these steps to shut down the system from the command line (including when you shut down from a Console Emulator login session):

1 Log in to the `root` account and enter

```
shutdown
```

The system responds by printing the `/etc/motd` file, the date, and a prompt that asks whether you want to enter a delay time other than the default of two minutes.

```
SHUTDOWN PROGRAM
```

```
Wed Oct 16 03:04:45 1991
```

```
Do you wish to enter your own delay (y or n)
```

2 Press RETURN or `n` to accept the default of two minutes, or press `y` to enter your own delay.

If you press `n`, the system waits two minutes before logging users off the system.

If you press `y`, the following prompt is displayed:

```
Enter your delay in minutes:
```

3 Enter 0 to start an immediate shutdown or enter the number of minutes to elapse before shutdown begins.

When you enter a delay time, you are prompted for a shutdown message:

```
Do you wish to enter your own message (y or n):
```

4 To print the default message, enter n; to print your own message, enter y.

If you enter n, the shutdown program broadcasts to all users:

```
Broadcast Message from root (console) Wed Jan 17 03:23:48
The system 'localhost' is going down in 2 minutes.
```

where *localhost* is your system name. If you enter y, you then enter your own broadcast message. Enter the text of your message and then an end-of-file character (CONTROL-D).

Your message might be something like

```
The system will be down for an hour for routine maintenance.
```

5 The system periodically broadcasts countdown messages. At the final minute before shutdown, you are asked whether you want to continue:

```
Do you want to continue (y or n):
```

Enter n to abort the shutdown procedure.

Enter y to put the system in single-user mode, which provides you with a terminal emulator interface. When the root prompt is displayed, you have the following options:

- Restart
- Shut Down the computer
- Remain in single-user mode in the console emulator
- Return the system to multi-user mode

6 To restart, enter reboot which returns the system to the Macintosh OS. If you then start up A/UX, the kernel restarts.

7 **To shut down the computer, enter `powerdown`, which automatically turns off the power to the CPU and console. You may then turn off the power to any external hard disks, in any order.**

8 **To return to multi-user mode and have the login dialog box displayed, enter `init 2`**

In this case, the same kernel remains in effect.

△ **Important** The `reboot` and `powerdown` commands from the command line assume you have performed the `shutdown` command first. Never use the command line version of the `reboot` and `powerdown` commands in multi-user mode. If you `reboot` or `powerdown` from a CommandShell in multi-user mode, the system does not perform all the cleaning up preferred for a proper shutdown. △

The natural order of startup devices

When the computer starts up, it looks for a Macintosh system folder on each device, in a specific order and in specific places. This information may be of interest when you consider changing the startup device. The computer looks in the following places in the following order:

1. Internal floppy disk drive number 0
2. Internal floppy disk drive number 1 (if it exists)
3. Hard disk with the highest SCSI ID number
4. Hard disk with the next highest SCSI ID number

The order continues down the chain of SCSI devices, each with a smaller SCSI ID number than the one before it. You can override the natural order by setting the hard disk from which you want to start as the startup device. The startup device is set from the Startup Disk control panel.

Changing the A/UX startup device

As shipped, A/UX Startup is configured to start the A/UX kernel from the same disk that contains A/UX Startup. You can change this to have A/UX Startup start the A/UX kernel from another device, which must be identified as a SCSI device with an ID from SCSI ID 0 through 6. Note that the disk is identified by its SCSI ID rather than by name. The A/UX kernel must be in the root file system of the disk you select as the boot device. Figure 2-6 shows how a system setup may look in this custom configuration.

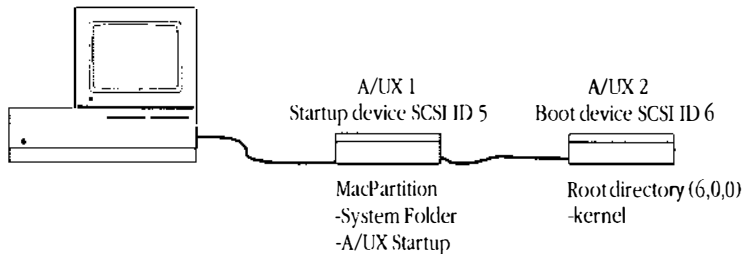


Figure 2-6 Changing the A/UX startup device

To change the A/UX startup device to be different than the device on which the application A/UX Startup resides, follow these steps:

- 1 Start the A/UX Startup application.**

Double-click the A/UX Startup icon, usually on the MacPartition disk.

- 2 Cancel the A/UX Startup application; press COMMAND-PERIOD.**

If A/UX Startup is password protected, you must have the `root` password to cancel the application and use the A/UX Startup commands.

- 3 Choose the General menu item from Preferences in the menu bar.**

The dialog box shown earlier in Figure 2-4 appears.

4 Identify the SCSI ID of the hard disk that contains the root file system and kernel.

Initially, the Root Directory box contains the `(default)/` parameter, which is the disk on which A/UX Startup resides. Change this parameter only when the root file system is on a different disk than A/UX Startup, as shown in Figure 2-6.

If your system is set up with A/UX Startup and the A/UX root file system on different drives, then you must change the `(default)/` value. You can use the `default` command in the A/UX Startup command shell window to find out which SCSI ID `(default)/` refers to.

5 Change the `(default)/` parameter in the Root Directory box.

Enter the SCSI ID number of the device you want to use as the A/UX startup device; use the following format where *ID* is the SCSI ID number:

```
(ID, 0, 0) /
```

In the example shown in Figure 2-6, you would enter

```
(6, 0, 0) /
```

On SCSI drives that have an external SCSI ID selector, you can verify the SCSI ID number by checking the indicator on the back of the external hard disk.

On SCSI drives that are assigned a SCSI ID through software switches, you must use the manufacturer's disk formatting or diagnostic utilities to discover the SCSI ID.

6 Click the OK box.

7 Start A/UX by choosing Boot from the Execute menu.

Your system now starts up A/UX from the disk configured as the startup device.

3 User and Group Administration

The A/UX working environment / 3-3

Administering user accounts / 3-10

Administering groups / 3-24

Files, directories, folders, and permissions / 3-27

How UNIX permissions and Macintosh file-sharing permissions compare / 3-35

Security / 3-38

A number of factors, which comprise the “user’s working environment,” affect how the Macintosh computer interprets and executes commands issued by an A/UX user. If you are an experienced administrator you might know these factors are the user’s file access permissions, group identification, login shell, and other UNIX environment variables typically set in what are referred to as the “dot files”—`.login`, `.profile`, `.cshrc`, and `.kshrc`. When setting up a user account, the administrator defines the user’s working environment.

Creating and maintaining “groups” is another administration task that increases in importance as the number of users who access the system increases. In installations where file security is an issue, the group concept is an effective way to ensure that access to files is limited to only those users who need to access them. Before proceeding with this chapter, you should be familiar with user and group, as well as the permissions, information that is introduced in *A/UX Essentials*.

The A/UX working environment

Within the A/UX system, each login account can define the environment in which it works. Each login account provides the following features:

- **A secure place to work:** When users want to use the A/UX system, they must first enter their login names and passwords. After logging in, an alias to the user's home directory folder is on the desktop (unless the user is `root`, or the folder has been put away using the Put Away item on the File menu). This arrangement permits a private environment in which each user has control over who has access to his or her work.
- **The ability to share tools and data:** The A/UX system also provides mechanisms by which users can share their work with other users. This is done mainly through the formation of groups of users with common tasks. Through the prudent use of groups, you can set up environments that permit the appropriate mix of security and sharing of resources.
- **The ability to customize:** Users can modify many features of their working environments by making changes to specific files located in their home directories. This gives them the power to personalize their environment. The system administrator, however, is responsible for the initial setup and certain kinds of modifications.

An introduction to the working environment

A number of factors determine the interpretation and operation of commands given by a user logged in to an A/UX system. This manual refers to these factors as the *user's working environment*. Additional terms that are used throughout this chapter to describe user working environment factors are described as follows:

- **Permissions:** Every file in a UNIX system has defined permissions that determine who can see, access, or change a file. UNIX permissions treat program files, data files, input and output devices, and even directories merely as “files.” Three actions can be performed on files: read, write, and execute. These three permissions can be set differently for three types of users: the user who created (or owns) the file, users belonging to the same group as the group affiliation of the file, and all other users. Users can execute programs or have access to files (including directories) only when specific permissions are granted.

Permissions in the A/UX Finder environment (See Files, See Folders, and Make Changes) are conceptually similar, though not identical, to UNIX permissions. For more information on how these two types of permissions compare, see *A/UX Essentials* and “How UNIX Permissions and Macintosh File-sharing Permissions Compare” later in this chapter.

- **Login name and password:** Before gaining access to the system, a user must enter a valid login name and password. Login names and the user’s initial shell are defined in the `/etc/passwd` file. (See “The `/etc/passwd` File,” later in this chapter.) If NIS is in effect, another file on a remote machine may be checked for a valid login and password, along with the `/etc/passwd` file. (See *A/UX Networking System Administration* for more information on NIS.)
- **User ID:** Each user login name is associated with a unique numeric user identification number (UID) that is recorded in the `/etc/passwd` file and identifies the user. The system administrator defines the UID. When a user attempts to use a command or gain access to a file, access is allowed or denied depending on the access permissions that are set on the file relative to the user’s UID. See *A/UX Essentials* for a discussion of file access permissions.
- **Group ID:** Each user login name is associated with at least one group identification number (GID). A user’s default, or login, group ID is defined in the `/etc/passwd` file, while the valid groups are defined in `/etc/group`. The group ID indicates the groups to which the user belongs at the time of login. Group membership is a security feature that permits a specific group of users access to files, while denying this access to users who are not members of the group. If NIS is in effect, another file on a remote machine may be checked for a valid group, along with the `/etc/group` file. (See *A/UX Networking System Administration* for more information on NIS.)

- **Home directory:** When users log in to the system, or execute the `cd` command without any arguments, they are placed in their home directory. This directory is defined as their home directory in the `/etc/passwd` file. Each login name should be associated with a home directory. (See “Administering User Accounts,” later in this chapter.) The specific location of the user’s home directory depends on how his or her system is set up, but a recommended location for all users’ home directories is in the `/users` (not `/usr`) directory. Users can customize their working environment by modifying setup files in their home directory.
- **Current directory:** The current directory is the user’s current location within the UNIX hierarchical directory structure. Every time the user changes to a new directory, the new directory becomes the current directory. When a user specifies a relative filename (a filename that does not start with a `/`), the current directory is searched for a file of that name.
- **Setup files:** Applications can have setup files in the user’s home directory. Usually setup files begin with a period. (These are also referred to as “dot files.”) For instance, you can have a `.mailrc` file to set your mail preferences and a `.xdefaults` file to set your X11 Window preferences.
- **System folders:** Each user may configure their own version of the Control Panels through the use of a personal System Folder. If an account does not have a System Folder in its home directory, the system-wide System Folder in `/mac/sys` is used.
- **Default shell program:** After logging in, the user needs some way of communicating with the system. The default shell program is the program that interprets user input (such as keystrokes) and directs output (for example, the contents of a file or words on the screen). The shell program is also called a command shell. (The term **command shell** is a generic term for several programs—C shell, Bourne shell, or Korn shell—whereas **CommandShell** is an application through which A/UX interacts with a command shell.) After a successful login, the shell program defined in the `/etc/passwd` file runs and executes a series of setup files that further define the user’s working environment. The setup files are discussed under “Command Shells and Setup Files,” later in this chapter. Common shell programs are `/bin/csh`, `/bin/sh`, or `/bin/ksh`. If this field in the `/etc/passwd` file is empty, the default shell program is the Bourne shell. The C shell is the default shell used by the `adduser` command. See `adduser(1M)`.

More restrictive programs can be used as the default shell program, for example `/bin/rsh` limits system access. This field can also be set up to execute a single command and then log out the user. See “Changing a User’s Shell,” later in this chapter, for more information.

- **Environment variables:** The shell programs provide a number of shell environment variables that can be assigned different values to alter the user’s working environment. Different shell programs have different variables available. Some environment variables are automatically assigned values from the `/etc/passwd` entry for each user. These include `LOGNAME` (login name), `HOME` (home directory), and `SHELL` (login shell program). Other variables, such as `PATH` (search path), `TERM` (terminal type), and `EDITOR` (preferred editor program) are assigned values when the shell prompt occurs or in files such as `.login`, `.cshrc`, `.kshrc`, or `.profile` in each user’s home directory.

Command shells and setup files

The setup files for the C shell—`/etc/cshrc`, `.cshrc`, `.login`, `.logout`—and for the Bourne and Korn shells—`/etc/profile`, `.profile`, and `.kshrc`—are discussed in this section.

The suggested default copies of these setup files are stored in the directory `/usr/lib/skel`. When a new account is created with the `adduser` program, these files are copied to the new user’s home directory. These files may be viewed or edited.

◆ **Note** When `$HOME` precedes a setup filename, as in `$HOME /.login`, it represents the user’s home directory. ◆

If you log in using 32-bit or 24-bit mode, your commands are processed by the Finder that operates in the A/UX environment. This Finder is different from the Macintosh Finder, but shares many of the same features. You can work directly from the A/UX command line interface rather than a Finder interface by selecting CommandShell from the application menu. This action creates a CommandShell window and displays an A/UX command line prompt. In these windows you enter traditional UNIX commands and receive responses from the system through one of several programs called command shells.

If you log in specifying the console emulator mode, a console window appears, and you bypass the Finder. The console emulator window will display an A/UX command line prompt and you are placed in your home directory. For more information about login modes, see *A/UX Essentials*.

Command line prompt

By default, the system host name is displayed in the command line prompt along with the user login name. The command line prompt is set by an entry in your default login script, which is `.profile` for Bourne shell or Korn shell users, or `.login` for C shell users.

If your system's host name is `picasso` and you have logged in as `root` (and you have not changed the contents of your login script), you will see the following prompt in your command shells:

```
picasso.root#
```

The C shell setup files

When a user logs in and `/bin/csh` is specified as the shell in the user's `/etc/passwd` entry, the system automatically runs several shell scripts before giving the user a prompt. The first of these scripts is `/etc/cshrc`. This is a script of shell commands that typically sets certain shell variables and sets a file creation mask (see "Setting Default File Permissions with `umask`," later in this chapter). This file is readable by all users but cannot be modified by normal users.

The `/etc/cshrc` file is run *before* the `.cshrc` file in the user's home directory. A user can override any definitions set by the execution of `/etc/cshrc` by redefining the variable in his or her own `.cshrc` or `.login`.

The `.login` script is run after `.cshrc`. It is typically used to set up terminal defaults and environment variables. After the initial login, whenever the user reinvokes the C shell program the `.cshrc` file is run again; `/etc/cshrc` and `$HOME/.login` are only run automatically at log in. Therefore, a user should place commands that need to be executed only once in the `.login` file in their home directory, and environment variables they want in all their shells in the `.cshrc` file.

Under the C shell, when a user logs out, the commands in his or her `.logout` file are executed.

The Bourne shell setup files

When `/bin/sh` is specified as the user's command shell in `/etc/passwd`, the system automatically runs several shell scripts before giving the user a prompt. The first of these scripts is the file `/etc/profile`. Similar to the `$HOME/.profile` file, this is a script of shell commands, which typically exports certain shell variables and sets a file creation mask. This file is readable by all users but cannot be modified by normal users.

The `/etc/profile` file is run *before* the file `.profile` in the user's home directory and thus serves as a default `.profile`. A user can override any definitions set by the execution of `/etc/profile` by redefining the variable in his or her own `.profile`.

The Korn shell setup files

When `/bin/ksh` is specified as the user's command shell, the system runs the `/etc/profile` and `$HOME/.profile` files described previously. If one of these files sets the `ENV` variable to any filename (`ENV` equals `$HOME/.kshrc` in the standard distribution), the system also executes the contents of the named file. After the initial login, whenever the user starts a new shell program (usually by opening a new CommandShell window), the file named in `ENV` is run again. (Note that `/etc/profile` and `$HOME/.profile` are not rerun.)

Macintosh System Folders

Users can create their own System Folders within their home directory folders by using the `systemfolder` program. If a user does not have a System Folder in his or her home directory, the default global System Folder `/mac/sys/System Folder` is used. While individual System Folders consume disk space, some reasons for a user choosing to create a personal System Folder could be:

- the user wants to maintain his own file sharing settings (export list), and users and groups file
- the user wants to customize his desktop, including fonts and inits
- the user does not want to work in the environment set by the global System Folder

How A/UX establishes the environment

A close look at a successful login will help you understand how different elements interact to determine a user's environment.

1. After successfully supplying the system with a valid login name and password, the `login` program reads the user's *UID*, *GID*, and *home-directory* fields in the `/etc/passwd` file. Next it invokes `initgroups`, which reads each line of the `/etc/group` file, looking for a match between the user's login name and the *login-name* field in this file. For each match, the user is assigned the corresponding group specified in the *GID* field. The `login` program then executes the program named as the user's default shell in the *startup-program* field of the `/etc/passwd` entry. This program inherits the user's user ID, group ID(s), and home directory from the `login` process.
2. The default shell program's first invocation is known as the *login shell*. Login shells look for and (if it exists) read a file in the directory `/etc` that contains commands to be run when the user logs in. If the login shell program is the C shell, the file is `/etc/cshrc`. If the shell is the Bourne shell or the Korn shell, this file is `/etc/profile`. In this file, the system administrator can modify certain aspects of all the users' working environments, such as `PATH`, `HOME`, and `TERM`; as well as set other features, such as aliases in the C shell and functions in the Bourne shell. See `sh(1)`, `ksh(1)`, and `csh(1)` in *A/UX Command Reference* for more information on the features of these shells.
3. After reading the default command file, the shell looks for and (if it exists) reads a setup file in the directory named in the *home-directory* field of the entry in `/etc/passwd`. If the default shell program is the Bourne shell or the Korn shell, the file `.profile` is executed; if the default shell is the C shell, both `.cshrc` and `.login` are executed. In these files, the user can modify any shell specific features of their particular shell as defined in the manual pages for each shell.
4. If the user is logging in to the A/UX 32-bit mode environment, the login shell executes the `.mac32` file in the users home directory. If that file does not exist, the file `./mac/bin/mac32` is executed. For A/UX 24-bit mode these files are `.mac24` and `/mac/bin/mac24`, respectively. For more information on which Finder environment to choose, see *A/UX Essentials*.
5. Once the startup environment has been established, the user may begin working.

Administering user accounts

The A/UX system administrator is responsible for providing adequate security for the users and information. The default security administered by the `adduser` program should prevent other users from reading or writing to a new user's area. Individual users may, of course, override this initial setup, but relaxation of security is an option, not the default. To ensure security, the administrator should understand the entries in the `/etc/passwd` and `/etc/group` files.

The `/etc/passwd` file

Each user's account is specified by a single entry in the `/etc/passwd` file. You must be `root` to modify this file.

The `/etc/passwd` file distributed with A/UX has several administrative logins and the user logins `start` and `Guest`. See "Administrative Logins and Groups," in Chapter 1, for a description of each of the default `login` accounts. See "Security," later in this chapter, for help in keeping these accounts secure.

◆ **Note** The `/etc/passwd` file on systems that use Network Information Service (NIS, formerly called Yellow Pages) contains additional information. See *A/UX Network System Administration* for details. ◆

Format of `/etc/passwd`

Each entry in `/etc/passwd` consists of one line with seven fields separated by colons.

The following is an `/etc/passwd` entry for a user named Toby Hobbes:

```
toby::1001:1001:Toby Hobbes,Rm 1,x12,5551212:/users/toby:/bin/csh
```

For more information on this section, see `passwd(1)` in *A/UX Command Reference* and `passwd(4)` in *A/UX Programmer's Reference*. The form of an entry is

login-name: password: UID: GID: misc: home-directory: startup-program

where the fields are interpreted as follows:

<i>login-name</i>	The name the user must use when logging in. It must be unique in the <code>/etc/passwd</code> file. This name should be no longer than eight characters. Typically the name is indicative of the person who uses it, such as <code>susan</code> or <code>bobw</code> .
<i>password</i>	An encrypted version of the actual password the user must use when logging in. The encrypting is done automatically when the password is first assigned and whenever it is changed. If the field is empty, as shown in the above example, no password is necessary to log in to that user account.
<i>UID</i>	A unique user identification number for each user.
<i>GID</i>	The user's default group identification number. Even if the user is listed in several groups in the <code>/etc/group</code> file (see "The <code>/etc/group</code> File," later in this chapter), he or she belongs by default to the group whose number appears in this field in the <code>/etc/passwd</code> file.
<i>misc</i>	Miscellaneous information about the user, such as full name, office, and telephone numbers.
<i>home-directory</i>	Whenever the user logs in to the system, this is the directory in which he or she is initially located.
<i>startup-program</i>	The name of an executable program, usually one of the command shells, that permits the user to communicate with the system.

Adding a user

Adding a user to your system is a two-step process: first, plan the user's working environment and second, add the user according to your plan. The planning stage is important; neglecting it can cause both security and administrative problems.

After you plan the new user's working environment, you can use the `adduser` command to easily add a user. The `adduser` command creates the entry in `/etc/passwd`, creates or copies the necessary directories and files, and sets access permissions for the new user. It is also possible to perform all these steps manually and add a user without using the automated script.

The recommended planning steps are presented first, followed by the manual procedure. For instructions on using the `adduser` Commando dialog box, see *Setting Up Accounts and Peripherals for A/UX*. If you would like to use the `adduser` command in CommandShell, refer to `adduser(1M)`.

◆ **Note** The `adduser` command does not permit new users to be added locally to a system that receives its password file through NIS. For adding users when NIS is active, see *Networking Essentials*. ◆

Planning a user's working environment

Follow these steps in planning a user's working environment:

1 **Keep a hard-copy record of data about the new user.**

The record should include information such as that listed in the following form.

This form simplifies adding a new user's working environment and is a useful record to keep.

Date you add the user (month/day/year) _____

User's real full name _____

User's telephone number (office) _____

User's telephone number (home) _____

User's login name _____

User identification number (UID) _____

Group identification number (GID) _____

Group name _____

Full pathname of the user's home directory _____

Full pathname of the user's default shell program _____

2 Record the full name and telephone number(s) of the user, along with the date you add the user.

You can choose to use either one or both of the home and office numbers. This information becomes available to `finger(1)`, which reports this information for a queried `login` name.

3 Pick a `login` name for the user.

Login names usually consist of all lowercase alphabetic characters. While a maximum of 15 characters are allowed for a local login, it is recommended you use only 8 characters for maximum compatibility. An 8 character `login` name is the maximum allowed for remote logins; if you always use a maximum of 8 characters for your `login` names, you will avoid conflict later if one day you want to connect your system to a network.

If you use the `adduser` command, it will check if the `login` name you have chosen has already been used. To make sure that the new user's `login` name is unique when configuring the accounts manually, enter the command

```
cat /etc/passwd | awk '{FS=":"; print $1}' | sort | more
```

If you are using NIS, use the command

```
ypcat passwd | awk '{FS=":"; print $1}' | sort | more
```

The output of this command will provide, in alphabetical order, a list of all the `login` names in use. Check if the name you wish to use is on the list. If the name has already been used, pick a different `login` name. When you decide on an unused name, record it on the hard copy form in the “User's `login` name” field.

4 Select a user identification number.

The `adduser` command will choose a new UID for you, or you can choose one yourself. Before selecting a new user identification number, you must find one that is not being used. One method for selecting the lowest unused number is to enter the command

```
cut -f3 -d: /etc/passwd | sort -n
```

This displays the current user ID numbers already used in the `/etc/passwd` file. Pick a number that is not being used and write it in the space labeled “User identification number” on your hardcopy report. By convention, ID numbers under 100 are reserved for special uses, such as for special system functions. The `adduser` command chooses numbers starting at 1000, by default.

5 **Select a group identification number.**

The `adduser` command will, by default, put each new user into a newly created group, but you can also specify a group ID to which you designate a user. See “Administering Groups,” later in this chapter, for information about planning, selecting and specifying group membership.

6 **Select a home directory.**

You always want to use absolute, rather than relative, pathnames when you enter paths into `/etc/passwd`. By default, `adduser` will put a user’s home directory into `/users`. Therefore an account with login name `toby` would have a home directory of `/users/toby`.

7 **Select a default shell program.**

You may want to ask if the user has a preference. If they do not have a preference, or aren’t sure of the differences between the C shell, the Bourne shell, and the Korn shell, choose the C shell. In that case, you would enter `/bin/csh` on this part of the record. See “Changing a User’s Shell,” later in this chapter, for information about using different command interpreters as a user’s default shell program.

Adding users manually

Although using the `adduser` command is far easier than adding a user manually, you may want to understand the steps involved for when you need to modify the `/etc/passwd` file by hand. This needs to be done when making changes to already existing accounts.

This information is sufficient for you to manually create all the files and entries required to add a user to the system.

1 **Log in as root.**

2 **Make a copy of** `/etc/passwd`.

For example,

```
cp /etc/passwd /etc/passwd.old
```

The `/etc/passwd.old` file is your backup in case you accidentally destroy this critical file.

^ **Important** Be careful while you modify this file. It is essential to your users' and your own ability to gain access to the system. /

3 **Edit the file** `/etc/passwd`.

Enter the command

```
vipw
```

This command invokes the `vi` editor and sets the `/etc/passwd` file as “read-only.” The contents of the password file are copied into a temporary file (`/etc/ptmp`). After you edit and write the file, the editor copies the changes back to the `/etc/passwd` file. The `vipw` editor locks the file so that it can't be modified by `passwd(1)` while `vipw` is in use.

◆ **Note** If you have the `/etc/passwd` file locked with `vipw`, another user with root permissions may still edit the `/etc/passwd` file with the `vi` command. For `vipw` to be effective, anyone with permissions to change the file must always use the `vipw` command ◆

For more information about using `vipw` to edit `/etc/passwd`, see `vipw(1M)` in *A/UX System Administrator's Reference*.

4 **Add the new user's information to the password file.**

Enter the following as the last line in the file, replacing each italicized word with the information for the user you are adding.

```
login-name: * : UID: GID: misc: home-dir: startup-program
```

Enter * in the second field (the password field) for now. It will be filled by an encrypted version of the user's password when the `/bin/passwd` command is run. The fifth field, *misc*, is of the form

```
Full name, Office, Office phone, Home phone
```

and is for use with the `finger` command.

Remember to use full pathnames, pathnames which start from `/`, for the user's home directory and default shell program. That is, use `/users/toby` instead of `users/toby` or `toby`.

Save the file and quit the editor.

5 **Set a password for the user.**

Enter the command

```
passwd login-name
```

where *login-name* is the name you entered in the first field of the new entry in the `/etc/passwd` file. You are asked to enter the new user's password. The `passwd` program asks you to enter the password twice. If you do not type the same password, it asks you to try again. If the password does not meet the systems requirements, it asks you to enter a different password (see `passwd(1)`). Reveal the password only to the new user, who should log in and set a new password as soon as possible.

6 **Create the user's home directory, using the pathname you entered to the sixth field (*home-dir*) of the new entry in the `passwd` file.**

Enter with the command

```
mkdir home-directory
```

for instance, it could be

```
mkdir /users/toby
```

7 **Copy the standard command files from** `/usr/lib/skel:`

It is a good idea to copy all these files, in case the user wants to sometimes temporarily change shells. Enter the following commands.

```
cp /usr/lib/skel/std.login home-dir/.login
```

for example,

```
cp /usr/lib/skel/std.login /users/toby/.login
```

Also,

```
cp /usr/lib/skel/std.cshrc home-dir/.cshrc
```

```
cp /usr/lib/skel/std.logout home-dir/.logout
```

```
cp /usr/lib/skel/std.profile home-dir/.profile
```

```
cp /usr/lib/skel/std.kshrc home-dir/.kshrc
```

```
cp /usr/lib/skel/std.profile home-dir/.profile
```

Note that basic copies of suggested `login` and environment files needed for each of the UNIX command shells are located in `/usr/lib/skel`. Use your own standard files if you have them, or edit these to your tastes.

8 **Change the ownership of the user's home directory and** `login` **or environment file or files.**

Again, replace each of the italicized words with the information you entered in the `/etc/passwd` file. Enter the commands

```
chown login-name home-directory
```

```
chown login-name home-directory/login-files
```

where *login-files* are the files you copied from `/usr/lib/skel`. For instance,

```
chown toby /users/toby
```

```
chown toby /users/toby/. [a-z]*
```

The notation `(. [a-z]*)` is a shorthand notation for all the files in `/users/toby` that begin with a period, and the second letter is any small letter of the alphabet. That is, it accesses all the files copied in step 7.

9 Change the group membership of the user's home directory and environment.

Enter the commands

```
chgrp group-name home-directory  
chgrp group-name home-directory/login-files
```

where *group-name* is the name of the group ID specified in the *GID* field of the user's entry in the `/etc/passwd` file. The group names associated with the various group IDs are recorded in `/etc/group` file.

For example,

```
chgrp mktg /users/toby  
chgrp mktg /users/toby/. [a-z]*
```

10 Change the permissions associated with the user's home directory and startup files.

Enter the commands

```
chmod 750 home-dir  
chmod 640 home-dir/.[a-z]*
```

The number 750 grants the user (owner) read, write, and execute permissions, grants members of the user's group read and execute permissions, and denies all permissions to other users. The number 640 grants the user read and write permissions, grants members of the user's group read permission, and denies all permissions to other users. See "Changing File Permissions: `chmod`," later in this chapter, and `chown(1)` and `chmod(1)` in *A/UX Command Reference*.

For example,

```
chmod 750 /users/toby  
chmod 640 /users/toby/. [a-z]*
```

This completes the steps for manually adding a user and you can log out.

Have the user log in using the new `login` name and password and create a new file in the working environment you just established.

Removing users

Removing a user from your system can be as simple as inserting the word `VOID` in that user's password field in `/etc/passwd`. However, if the user has created many files that must be saved, you may need to find all files owned by the user, back them up, examine each of them, determine who else uses the files, change the ownership of shared files, remove links, and finally delete the user's password entry.

This section introduces the simplest form of user removal first and then discusses additional steps to remove all files and directories associated with that user.

Voiding a user account

The first step in removing a user from your system is to deny the user access to it. The best way to do this quickly is to edit the user's `/etc/passwd` entry and enter the word `VOID` in the second field, the password field. This makes it impossible for anyone to log in as that user, although that user's files remain unaffected.

◆ **Note** Do not leave the password field blank. A blank password field can result in a serious security breach; anybody can log in to the system through an account without a password. ◆

It is not a good idea to delete the whole `/etc/passwd` entry for the user yet. If you do, you affect the files owned by that user. Commands that use `login` names as arguments (for example, `chown` and `find`) or that print information relating to `login` names (for example, `ls -l`) check the `/etc/passwd` file for the user `login` and UID number. For instance, if you have deleted a user's entry, and then run `ls -l` on files owned by that user, there is no `login` name listed as a file's owner, instead it is replaced by the number which is the UID of the (now deleted) user. If you delete a few `/etc/passwd` entries, you may get confused about which files belonged to which former user.

Deleting a user account

In general, it is a good idea to back up a user's files before deleting them, for two reasons:

- These files may contain information that you will need later.
- These files may be used by other users on your system.

To properly delete a user account, follow these steps:

- 1 **Void the user's password.**
- 2 **Find all the files belonging to the user, regardless of their location, with the command**

```
find / -user login-name -print > filename
```

where *login-name* is the user's `login` name and *filename* is the name of a file. This command creates a file, named *filename*, which contains a list of all files on the system that are owned by the user *login-name*. Use this file as a reference while completing the following tasks.

- 3 **Back up the files using either `tar` or `cpio`, or drag them onto a floppy disk.**
See the information on `tar` and `cpio` in Chapter 5, "Backing Up Your System."
- 4 **Delete the user's files only after finding out if anyone is currently executing any commands or using any data files owned by that user.**
Inquire personally or through `mail` to find out if any others regularly use files created by that user. If they do, change the ownership of those files. If a file is linked to a file owned by that user, remove the link. Then delete the files.
- 5 **Alternatively, you may wish to use `chmod` and `chgrp` to assign ownership of these files to active users.**

Moving a user's home directory within a file system

Sometimes you must move a user's home directory and files. There are several ways of doing this, and the method you choose depends on the characteristics of the move. If you do move a user's home directory, remember to change the home directory field in the users entry in `/etc/passwd`.

The simplest move is the one that involves moving a user's directory to another place in the same file system. The command line

```
mv old-directory new-directory
```

moves the *old-directory* directory (including all of its files and subdirectories) to *new-directory*.

Moving a user's directory across file systems

In the UNIX operating system, you cannot use the `mv` command to move files from one file system to another. For example, while adding an additional hard disk to your system you create a new file system on that disk. If you wish to move existing user accounts to that disk, you may use one of the following methods, which allow you to move files from one file system to another. The `cp -r` command will copy files across file systems, but it also changes the owner of the files to the person that is running the `cp` command. The `cp -r` command is not recommended for moving a user's files across file system.

Using `cpio`

With `cpio`, which stands for "copy input to output," a directory containing files and subdirectories can be copied elsewhere on the system, with all files maintaining their original ownership, permissions, and modification time.

◆ **Note** In the standard A/UX distribution on a Macintosh computer with a 160 megabyte (MB) hard disk, the disk contains only one user-accessible file system: `/` (the root directory). The entire A/UX directory hierarchy and any specific hierarchy (such as `/users`) are available on this file system. Therefore, the `mv` command will work, and you do not need to use `cpio`. ◆

If you have created a new file system (for example, located at `/users2`) on an additional hard disk, you can copy all files and subdirectories contained in the directory `/users/toby` to a directory called `/users2/toby` on the other file system.

1 **Change to the `/users` directory.**

Enter the command

```
cd /users
```

2 **Enter the following command:**

```
find toby -depth -print | cpio -pdm /users2
```

This command copies the user's files across file systems to a new directory. Once you are sure that the copy was successful, you can delete the original files.

This example shows how to move the user and is not a lesson on `cpio`; see `cpio(1)` in *A/UX Command Reference*. Remember that when you move the user's files, you should also change the user's *home-directory* field in `/etc/passwd` and any other references to his or her home directory in startup files such as the users `.profile` or `.cshrc` files.

Using `tar`

The `tar` (tape archiver) command can also be used, instead of `cpio`, to copy directories from one file system to another.

◆ **Note** In the standard A/UX distribution on a Macintosh computer with a 160 MB hard disk, the disk contains only one user-accessible file system: `/` (the root directory). The entire UNIX directory hierarchy and any specific hierarchy (such as `/users`) are available on this file system and you need not use `tar`. ◆

If you have created a new file system (for example, one located at `/users2`) on an external hard disk—or even a floppy disk—you can copy all files and subdirectories contained in the directory `/users/toby` to a directory `/users2/toby` on the other file system.

1 **Change to the** `/users` **directory.**

Enter the command

```
cd /users
```

2 **Enter the following command:**

```
tar cf - toby | (cd /users2; tar xf -)
```

This command uses `tar` to copy the user's files across file systems to a new directory. Once you are sure that the files are copied successfully, you can delete the original files.

This example shows how to move the user and is not a lesson on `tar`; see `tar(1)` in *A/UX Command Reference*. Remember that when you move the user's files, you should also change the user's *home-directory* field in `/etc/passwd` and any other references to his or her home directory in files such as `.profile` or `.cshrc..`

Changing a user's shell

The last field in the user's entry in the `/etc/passwd` file determines the user's default startup program. Typically, the field is `/bin/csh` (the default when using `adduser`), `/bin/sh`, or `/bin/ksh`, (for the C shell, Bourne shell, and the Korn shell, respectively). If there is nothing in an `/etc/passwd` entry after the sixth colon (:), the default shell, `/bin/sh`, is used.

To change a user's default startup program, you need only change this field. Either `root` or a user can execute the change shell command, `.chsh(1)`, to change the default startup program to any legal value, as listed in `/etc/shells`. (A user can only change his or her own default shell, `root` can change any user's shell.) Since the `chsh` command actually changes the entry in the `/etc/passwd` file, if this command is used other modifications to the user's working environment may be necessary, particularly with regard to shell startup files in the user's home directory.

Any program at all can serve as the default startup program. For instance, the last field of the `/etc/passwd` file can be a program such as `/bin/who`. If `/bin/who` is the default startup program, the account allows for a user to log in, view the output of the program `/bin/who` and then the user account is logged out automatically.

Although `/bin/who` is not a very useful working environment, other programs, such as the restricted shell, `rsh`, may be. See `sh(1)` for more information on `rsh`. The `rsh` program allows a user the use of a shell within the home directory but allows no directory changes.

Administering groups

Defining groups of users on a system provides a way of combining user access permissions for people who have the same interests or responsibilities. Before you add a user, you should have a clear idea of what his or her tasks will be, what other users are currently engaged in similar activities, what parts of the system you want the user to have access to, whether a new group should be created, and where in the system the new user should be located.

In other words, the new user should belong to a group whose members have similar tasks (accounting, legal, programming, documentation, and so on), or to more than one group if the user will have a variety of tasks.

Creating groups

When you administer several users on a system, you will want to create groups that correspond to the user's activities. For example, group names might include examples such as `legal`, `eng`, `pubs`, `acctg`, `mkt`, and so forth. Group names are arbitrary, but usually related to the group's activity.

When you create groups, you add an entry to the `/etc/group` file that includes a group ID number. Group ID numbers can be any value you wish, but there are some practical guidelines you can consider:

- Create group numbers in round hundreds or thousands as shown in this example of `/etc/group` entries:

```
mktg:*:300:toby,susan,ng
eng:*:5000:fred,toby,seth
```

- When you add users to the `passwd` file, relate their user ID (UID) to the group they are in. For example,

```
toby:aX3eopRx:301:300::/user2/mkt/toby:/bin/csh
susan:ZzXpL3eox:302:300::/user2/mkt/susan:/bin/csh
fred:7IoiudSSe:5012:5000::/user2/eng/fred:/bin/ksh
```

By convention, user ID numbers under 100 are reserved for special uses, such as for special system functions; therefore, the group ID number chosen should follow the same guidelines. The `adduser` command chooses numbers starting at 1000, by default. You may specify to `adduser` which number to use for both a user and group ID.

By organizing your groups and users systematically, your system will be easier to administer as it grows and expands.

The `/etc/group` file

The GID field of a user's entry in the `/etc/passwd` file establishes a single default group for the user. The `/etc/group` file is used to establish multiple group memberships for a user.

The `/etc/group` file contains entries with four fields separated by colons. The form of the entry is

group-name:password:GID:list

where the fields are interpreted as follows:

<i>group-name</i>	The group name. Group names are arbitrary, but by convention their meanings should be self-evident (for instance, <code>acctg</code> rather than <code>xyz24</code>). If this field is a "+", NIS is controlling the <code>/etc/passwd</code> file. See <i>A/UX Networking System Administration</i> for information on NIS.
<i>password</i>	An encrypted version of the password for this group. Traditionally this field is not used in UNIX. There is no standard software to give groups passwords. It is common practice to disable group password checking by filling the <i>password</i> field with the word <code>VOID</code> or with asterisks (*). Setting proper group permissions on files and directories makes this field superfluous. However, you should not leave this field blank, as that will allow anyone to use the <code>newgrp(1)</code> command to become a member of this group.
<i>GID</i>	A unique number set for each group. For each group ID there is only one group name, and vice versa. The actual group ID numbers that exist in the <code>/etc/group</code> file are the only numbers that should be entered in the <i>GID</i> field of <code>/etc/passwd</code> entries. The group ID entered for each user in <code>/etc/passwd</code> should coincide with the group ownership of that user's home directory.
<i>list</i>	A list of the <code>login</code> names of the members of the group. The <code>login</code> names must be separated by commas. Entering a user's <code>login</code> name in the group's <i>list</i> field is optional for those users who belong to only the group indicated in the <i>GID</i> field of the <code>/etc/passwd</code> file.

Multiple group memberships

It is possible for a user to be a member of several groups. When a user who is a member of more than one group creates a file or directory, the group associated with the parent directory becomes the group associated with the new file or subdirectory. You will notice in the earlier example that the user `toby` is listed in both `mktg` and `eng`. If the group ownership of his home directory is `mktg`, then every file he creates and every directory he makes below his home directory will also belong to `mktg`. For various reasons, he might want some of the files he creates to belong to the group `eng`. There are two ways he can accomplish this:

- He can create a file, which will have group ownership `mktg` by default; he can then change the file's group membership to `eng` after the fact.
- He can create a directory which will have group ownership `mktg`; he can then change the group membership of the directory to `eng`. Every file created within that directory will then also belong to `eng`. (This is true in UFS file systems only; in SVFS file systems, the new file will take on the group of the creator.)

The way A/UX handles groups is derived from the method used by the Berkeley version of the UNIX operating system, and this method differs from the way System V of the UNIX operating system handles groups. (In the System V version, the user is allowed to be in only one group.) In A/UX, a user can be in a maximum of eight groups. The system administrator enters the groups to which a user belongs into the file `/etc/group`. To list your group memberships, enter the command `groups`.

If a user belongs to eight groups and temporarily needs to be in yet another group, he or she must enter `newgrp groupname`, where `groupname` is an entry in `/etc/group`. In order to use the `newgrp` command, you must be listed as a member of the group in `/etc/group`, or know the password of the group. This causes `groupname` to replace the first group listed in your environment for the duration of the `login` session. Note that a user's group membership is still restricted to eight groups. The `newgrp` command temporarily substitutes the new group in place of the first group in your internal list.

Files, directories, folders, and permissions

By default, security for accounts set up with the `adduser` command prevents other users from reading or writing to a user's area.

UNIX file-access permissions

For an introductory discussion of permissions, and information on how they are represented in the Finder, see *A/UX Essentials*. The following section discusses the A/UX command shell representation of permissions.

File permissions

A/UX Essentials explains that UNIX file access is designated by read, write, and execute permissions for owner, group, and others.

When displaying permissions using the `ls -l` command, the output of this command might show

```
-rwxr-xr-- toby eng 512 Oct 3 17:51 importantfile
```

The file access permissions that appear at the left side of the display can be interpreted as shown in Figure 3-1. For the file `importantfile`, the user has read, write, and execute permission; members of his group have read and execute permission (but cannot change the file); all other users on the system may only read the file.

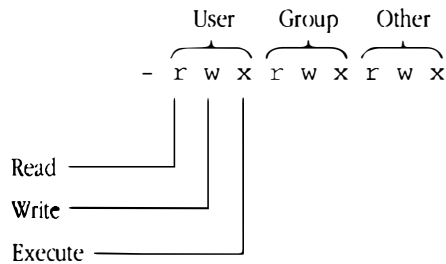


Figure 3-1 Access classes

The initial character, shown as a hyphen (-), represents the file type. Table 3-1 lists the various file types. For more information on different file types, see `chmod(1)`.

Table 3-1 File type characters

Type	Description
-	A regular file that contains data.
d	The file is a directory.
c	The file entry represents a character device.
b	The file entry represents a block device.
p	The file entry represents a named pipe or FIFO.
s	The file entry represents a socket.
l	The file entry represents a symbolic link to another file.

Directory and folder permissions

As explained in *A/UX Essentials*, when file access permissions are assigned to a directory, they work a little differently. Here is a list of directory and folder access permissions and their meanings:

- r Allows you to list the names of the files that are in the directory (`ls`).
- w Allows you to add or delete entries from that directory.
- x Allows you to search the directory, or to make it the current directory (`cd`).

Setting permissions on a directory affects only the directory itself and does not change the permissions settings of any of the directory's files or subdirectories.

Directory permissions are among the most important aspects of the user's environment. For example, *file* permissions that protect against reading or writing by other users are not enough to protect the file from being deleted, if the *directory* permissions allow other users write permission. Similarly, if the directory grants the group read permission, its files can be listed (`ls`) by a group member even if the files themselves deny group read permission and cannot be opened.

Group membership is an important consideration for the administrator setting up directory permissions. The default group membership of a file or directory is the same as the group membership of the directory in which the file is created. This allows for the creation of hierarchies of directories according to their group membership.

Directory permissions can affect the accessing of a file. If a wildcard (such as `*` or `?`) is used in the path specification, read permission will also be required for the affected directory. This is a result of the wildcard's causing the shell to read the directory (on the user's behalf) to find the requested file. Removal of read permission from directories can thus be used to prevent snooping, while allowing access to specific files.

Changing file permissions: `chmod`

Only the owner of a file, or root, can change a file's permissions using the `chmod` command. Anything that `chmod` can do to a file's permissions it can do to a directory's permissions as well, because A/UX treats a directory as a file. Note that the `chmod` command does not apply to files and folders on Macintosh file systems. The `chmod` command can be invoked with either symbolic or numeric terms.

Symbolic terms

Symbolic terms are straightforward: `u` stands for user (that is, owner) of the file, `g` stands for group, and `o` stands for others; `+` represents granting permission and `-` represents denying permission. The format for invoking `chmod` with symbolic terms is as follows

```
chmod access-class operator permissions filename
```

and consists of the following four arguments. Note that there are no spaces between the *access class*, *operator*, and *permission* fields in the examples below.

<i>access-class</i>	One or more of the three access classes—user (<code>u</code>), group (<code>g</code>), or other (<code>o</code>) as described in <i>A/UX Essentials</i> . In addition, the access class all (<code>a</code>) lets you grant or deny permissions to all three access classes simultaneously.
<i>operator</i>	Grants access permission (the <code>+</code> operator) or denies it (the <code>-</code> operator). You can't both grant and deny permissions in a single command. You must grant permissions to one access class in one command, then deny it to another access class in a second command.
<i>permissions</i>	Read permission (<code>r</code>), write permission (<code>w</code>), and execute permission (<code>x</code>). You can grant (or deny) more than one type of permission at the same time. Also includes <code>setuid</code> or <code>setgid</code> (<code>s</code>) and sticky bit (<code>t</code>), discussed later in this chapter in “Commands That Assume Permissions Using <code>setuid</code> and <code>setgid</code> .”
<i>filename</i>	The file or files whose permissions are to be changed. You may use absolute or relative pathnames.

For example, to change the permissions of a file, named *filename*, from

```
-rw-rw-rwx
```

to

```
-rwxrw-r--
```

the sequence of commands is as follows:

1 Grant execute permission to the owner:

Enter the command

```
chmod u+x filename
```


2 Deny write and execute permissions to all others:

Enter the command

```
chmod o-wx filename
```

Numeric terms

Numeric (or absolute) terms are based on octal numbers where, for each access class, the mode of the file is set according to its octal sum. It sounds complex but is rather simple if you understand basic addition. Figure 3-2 shows how you can easily translate the desired file permissions into numeric terms.

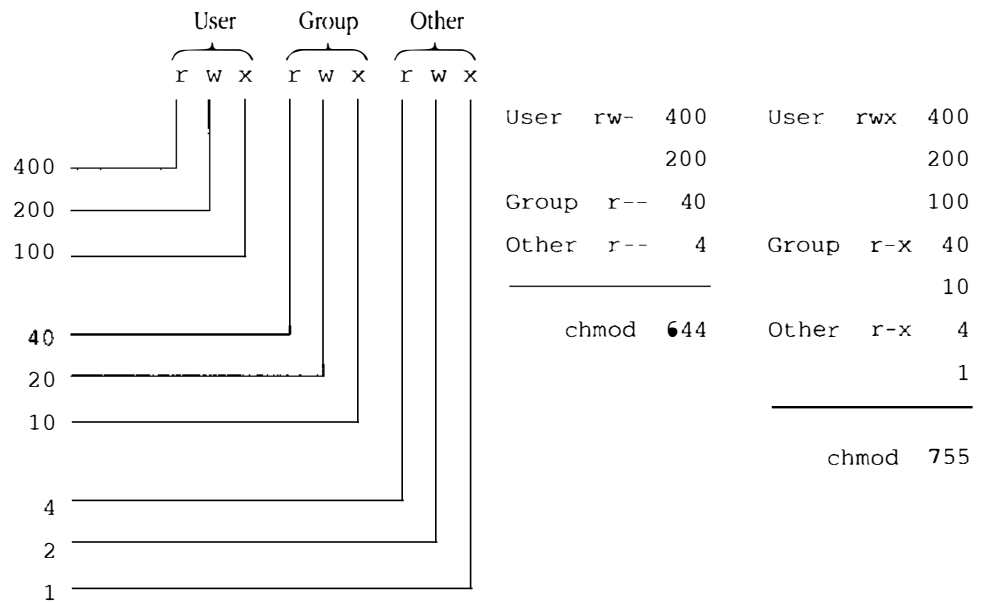


Figure 3-2 Numeric values for `chmod`

As Figure 3-2 shows, by adding up the values associated with each permission (r, w, x), you arrive at the numeric term to use with the `chmod` command.

The format for invoking `chmod` with numeric terms is

```
chmod permission filename
```

where *permission* is the numerical representation for each access class. For example, to assign a file read, write, and execute permission for the owner and to deny any access by group users or other users, you would use the following command:

```
chmod 700 filename
```

The permissions of the file are then

```
-rwx-----
```

Similarly, if you use the command:

```
chmod 754 filename
```

The permissions of the file become

```
-rwxr-xr--
```

Commands that assume permissions using `setuid` and `setgid`

It is possible to set up commands *to act as if* they were being invoked by a specified user or by a member of a specified group. The mechanism for this is called *set user identification* (`setuid`) or *set group identification* (`setgid`). The `setuid` permissions on a command allow the command to run with the permissions of its owner (the owner of the command that is being executed) rather than the normal behavior in which the command runs with the permissions of the person running the command. The `setgid` command functions similarly but runs with the permissions of the group affiliation of the command.

For example, the `passwd` command, which a user invokes to change his or her password, is a `setuid` program, `/bin/passwd`. When invoked, `/bin/passwd` takes on the identity of the owner of the `passwd` program, in this case `root`, for the time needed to modify the file `/etc/passwd`.

Of the two, commands using `setgid` permissions tend to be safer, since group membership typically confers less power. Both should be treated with care, however.

It is possible to set up a program that can be run with `setuid` permissions only by a selected set of users. You do this by putting the users into the same group to which the program belongs and denying execute permission to all other users. Then, only group members can run the program, performing the action as if they were the owner of the executable file.

You also use `chmod` to turn the `setuid` bit or `setgid` bit for a file on or off. The last section described using `chmod` with three numeric terms, such as `755`. The `setuid` or `setgid` bit is a fourth optional digit using `chmod` numeric terms. When using symbolic terms, use the corresponding characters in parenthesis below in the *permissions* field:

- 1 (t) Set sticky bit (not used by A/UX)
- 2 (s) Set GID bit on execution
- 4 (s) Set UID bit on execution

◆ **Note** `setuid` and `setgid` bits are applicable only with users or group permissions, not with permissions governing other users. Also, neither `setuid` nor `setgid` modes apply to directories or nonexecutable files. ◆

Turning on the `setuid` or `setgid` bit is useful with very specific and restricted files—for example, the `/bin/passwd` program. The `chmod` command that turns on the `setgid` bit on a file with read only permissions for all (mode `444`) is

```
chmod 2444 filename
```

The command to turn on the `setuid` bit on a file with read, write, and execute permissions for the owner, read and execute permissions for the group, and no permissions for all others (mode `750`) is

```
chmod 4750 filename
```

The permissions field in the output of the `ls -l` command in the first case is

```
-rwxrwsrwx
```

where the `s` in the group execution field represents the `setgid` bit.

The permissions field in the output of the `ls -l` command in the second case is

```
-rwsr-x---
```

where the `s` in the owner execution field represents the `setuid` bit.

You can combine the setting of the `setuid` bit and the `setgid` bit, as you can with all other numeric terms, so that

```
chmod 6755 filename
```

results in

```
-rwsr-sr-x
```

The sticky bit

In systems that load an entire file into physical memory, data is swapped in and out of memory as needed. These systems are called *swapping systems*. *Paging systems*, however, load a page (4 kilobytes [4K] in A/UX) of the requested data instead of a whole file at a time. This speeds data retrieval. A/UX is a paging system.

In swapping systems, the sticky bit indicates that the file should remain in main memory once it has been loaded in; this can shorten initialization time for frequently used programs at the cost of tying up a portion of main memory indefinitely. Because A/UX is a paging system, however, the sticky bit has no effect.

Setting default file permissions with `umask`

The `umask` command defines the default permissions for each file created by a user. You can run this command for all users in the `/etc/profile` or `/etc/cshrc` file, or each user can run it individually in his or her `.profile` or `.login` file. See “How A/UX Establishes the Environment,” earlier in this chapter.

The `umask` command, like the permissions associated with `chmod`, is assigned a numeric value of three octal numbers. The value of each specified digit of `umask` is *subtracted* from the corresponding digit specified in a completely open (777) file.

For example, to ensure that all files created by a user have the permissions

```
-rwxr-x---
```

you must set the `umask` for that user as

```
umask 027
```

Thus, when the `027` is subtracted from `777`, the files permissions are `750` (`-rwxr-x---`). The default `umask` in the A/UX standard startup files is `027` for regular users.

The notation

```
umask 27
```

is shorthand for

```
umask 027
```

That is, leading zeros can be eliminated from the notation.

Note that changing a user's `umask` does not affect the permissions on existing files. It only affects permissions on subsequent files created.

How UNIX permissions and Macintosh file-sharing permissions compare

A/UX Essentials introduces permissions in both the UNIX and Finder environments. This section discusses some further details about permissions that will help the system administrator understand the difference between Macintosh file-sharing permissions and UNIX permissions; before reading this chapter, you should be familiar with the information in *A/UX Essentials* and with Macintosh file-sharing permissions.

There are two sets of users that are able to access an A/UX system. The first set is the users that are able to log in because they have an entry in the `/etc/passwd` file. The second set are the users that are able to mount a volume and file share from the same A/UX system because they have access through an entry in the Users & Groups control panel. Likewise, there are two sets of permissions on an A/UX system—UNIX permissions, which are set via the `chmod` command (or UNIX Permissions menu), and file-sharing permissions, which are set via the Sharing menu. Although these two sets of users enter the system differently, and they appear to have separate sets of permissions, a system administrator must properly set access permissions for both types of users accessing the system.

Each user that logs in to A/UX and uses the Finder has the capability to share files on the system with other users over a network. If the A/UX user has a personal System Folder in his or her home directory, the settings in the Sharing Setup and Users & Groups control panels are specific to that user. Otherwise, these control panels settings are in the system wide control panel in `/mac/sys/System Folder`.

◆ **Note** If different users on your system use file-sharing, the only way to ensure that the file-sharing settings you configure remain the same between your `login` sessions is to have a personal System Folder. Otherwise, anyone who uses the global System Folder `/mac/sys/System Folder` may change the settings—and they may not be as you left them when you last logged in. ◆

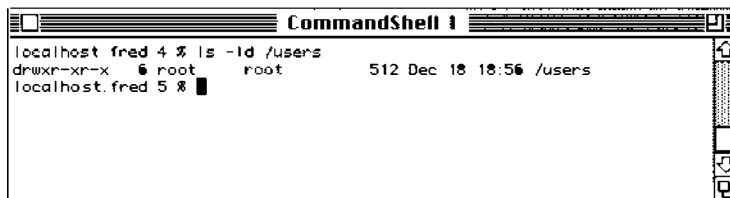
Since the A/UX Finder exists on the console, access to the A/UX system through file-sharing depends on the person logged in on the console, and the settings of the control panels in that person's System Folder. Furthermore, when that user logs out of A/UX, that file-sharing session stops as well.

Likewise, the folders that are shared and the access permissions associated with those folders are also correlate to the user logged in on the console. The permissions a user is allowed to grant when sharing a folder depends on the UNIX access permissions that the user is allowed.

For example, Figure 3-3 shows the UNIX permissions of the user `fred` on the directory `/users`.

As shown by both the `ls -l` listing of `/users` in Figure 3-3, and the Finder representation of `/users` in Figure 3-4, `fred` does not have write permission (equivalently, Make Changes permission in the Finder) in the `/users` directory. Therefore, when `fred` selects the `users` folder for file-sharing, the Make Changes permissions are not enabled. (See Figure 3-5.) Since `fred` does not have permission to make changes, he can not grant that permission to others through file-sharing.

Effectively, the way this works is that file-sharing permissions are an additional layer of permissions over the UNIX permissions. A user can only grant as much access to a folder for file sharing as that user possesses in UNIX. It is for this reason that it is important to understand the section in *A/UX Essentials* that discusses how UNIX read, write, and execute permissions compare with file-sharing's See Folders, See Files, and Make Changes permissions.



```
CommandShell 1
localhost fred 4 % ls -ld /users
drwxr-xr-x  6 root   root       512 Dec 18 18:56 /users
localhost.fred 5 %
```

Figure 3-3 CommandShell representation of permissions of `/users`

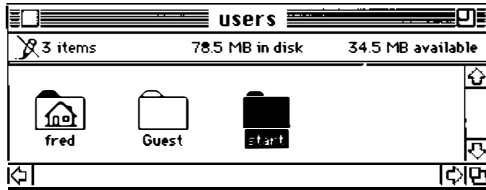


Figure 3-4 Finder representation of permissions of `/users`

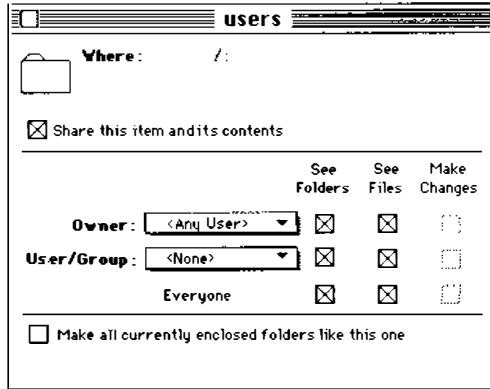


Figure 3-5 File-share permissions enabled for `/users`

The implications of this permission scheme is that if the system administrator has properly set access permissions for a user within A/UX, then the file-sharing permissions for that user are also properly configured. However, it also implies that root must be very careful which folders it shares. For instance, if root were to share `/etc`, and keep the default file sharing permissions, anyone could enter the system and change the `/etc/passwd` file.

◆ **Note** The file-sharing permissions existing as an additional layer of access permissions over A/UX permissions is only relevant in UNIX-based file systems. In HFS file systems, access through file sharing performs as expected under System 7. ◆

Security

This section offers guidelines to consider when implementing security on your system.

Know your open accounts

For purposes of this discussion, open accounts are those that allow virtually any user to log in. The two open accounts that A/UX supplies are Guest and start. Guest has no password by default and the start account has the default password `my.password` printed in the A/UX documentation.

The system administrator can take the following steps to ensure security on the open accounts:

- Set a password. Note that, without password aging, anyone who knows an account's password can change it and not tell you.
- Set a password, and use password aging so that only root can assign a new password.
- Disable the open accounts by putting the word `VOID` or an asterisk (*) in the password field of the Guest and start entries in `/etc/passwd`.

Monitor `/etc/passwd` and `/etc/group`

To monitor the `/etc/passwd` file, use the password check command, `pwck(1M)`. To monitor the `/etc/group` file, use the group check command, `grpck(1M)`; both these commands generate a report that show inconsistencies in these files.

Multiple users—one login

If you are looking to have one account be a generic account, for example, for a whole department, do not use one `login` name and give the password to different people. Use different `login` names and set the home directory and the shell of each `login` to the same values as the other related logins. That way, you have individual `login` names to refer to if you ever need to trace an event. You should also make a group in `/etc/group` under these conditions and put all the relevant logins as members of the group.

Password aging

Password aging is used to enforce rules and restrictions on a user's password. For example, you can configure the system so that only root, not the user, can change an account's password. Password aging is also useful if you want users to change their passwords regularly. Regularly changing a password can improve system security by preventing a surreptitiously obtained password from being used for very long. Password aging is described in detail in the `passwd(4)` manual page.

Password restrictions

As distributed, passwords in A/UX must satisfy several requirements. They must contain more than six characters, include a non letter, and not be a rotation of the user `login` name. These restrictions are part of the System V implementation of UNIX and are used in A/UX.

Permission restrictions

Programs that have the setuid bit set are potential security hazards. The `-s` option of `nccheck(1M)` reports those files that have this bit set. Periodically checking your system and comparing a list of files with the setuid bit set to a previous list can help discover possible security breaches.

4 Preparing a Hard Disk for A/UX

The steps in preparing disks for A/UX / 4-3

About UNIX file systems / 4-3

Background on Apple Hard Disk SC Setup / 4-9

Partitioning disks / 4-12

Making A/UX file systems / 4-29

Mounting A/UX file systems / 4-32

This chapter describes how to place a UNIX file system on a hard disk. Preparing a hard disk involves partitioning the disk to hold distinct types of data and creating data structures to organize that data. **Partitioning** a disk is a method of sectioning a single disk so you can treat it as though it were multiple disks. This allows you to collect files together and perform various tasks, such as backups, on the files as a group. In nearly all cases, partitioning is done with the Apple Hard Disk SC Setup program. The *SC* in the name stands for the interface that connects hard disks to Apple computers—**SCSI** (Small Computer System Interface). This chapter begins with a general discussion of the partitioning process. For detailed information on file system structure, see Chapter 8 “Checking the A/UX File System: `fsck`.”

For complete step-by-step instructions on the specific situations of using Hard Disk SC Setup to partition a disk for user files, a `/usr` partition, and for UNIX files and the Macintosh OS, see *Setting Up Accounts and Peripherals for A/UX*.

The steps in preparing disks for A/UX

In general, there are three steps to take in order to use a hard disk with A/UX. These steps are:

- Partition the disk—this process allows you to treat a single disk as though it were several disks, making the information in each partition easier to manage.
- Create a file system—this sets up organizational structures within a partition so the operating system can find files.
- Mount the file system —this process creates a link between an A/UX directory and a file system.

You can easily prepare A/UX disks by using Apple Hard Disk SC Setup 3.0, which guides you through all the above actions. You can further tune the performance of your disk by using Hard Disk SC Setup only to partition the disk and using other utilities to make and mount the file systems for that disk. The procedures for doing this are given later in this chapter.

⚠ **Important** Be sure to use Hard Disk SC Setup 3.0. Other versions do not have all of the support capabilities discussed in this chapter.

About UNIX file systems

This section provides background information on UNIX file system structures, and may already be familiar to experienced system administrators. Relatively new administrators may wish to read this section to better understand the workings of the operating system.

Disk partitions

With Apple Hard Disk SC Setup 3.0, you can subdivide a hard disk into logical sections called **partitions**. A partition is a contiguous region of a disk drive that is used to hold data. Partitions allow a single disk to accommodate multiple file systems and even

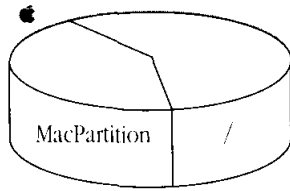


Figure 4-1 Two partitions on a disk

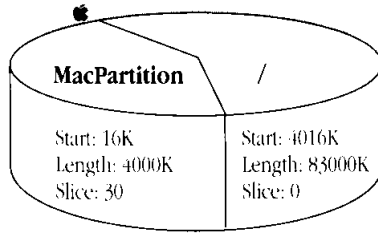
multiple operating systems because data from one partition cannot overlap into another partition. Figure 4-1 shows a disk containing one partition for a UNIX file system (called /) and one for a Macintosh Operating System (called MacPartition).

Storing data in separate partitions saves time when you make backups. By putting system files in one partition and user files in another, you can more easily manage them. When making backups you can concentrate on the partition holding the user files, which change often and thus need backing up more frequently than the system files. You can back up the partition that holds the system files, which seldom changes, less often. In general, backups can be administered more efficiently when files are thoughtfully distributed among disk partitions.

The system keeps track of partitions by specifying their locations in a map of the entire disk. This map is called a **partition map**, and is placed in a reserved area of the disk. In order to be located, each partition on a disk must have an entry in the disk partition map (DPM). A DPM entry includes the starting block, length, and name of each partition on the disk, as shown in Figure 4-2. Thus, the act of partitioning establishes the starting physical address of a partition and the number of blocks allocated to it. Usually A/UX partitions also include a **slice number**, which is a special index to the partition map used by the A/UX operating system.

Since a partition may be used for Macintosh file systems, or even other operating systems, A/UX prevents access to partitions that have not been identified for UNIX use. A/UX locates the partitions that could contain valid UNIX file systems, or otherwise be used by A/UX, by first reading the partition map.

A partition that has been identified for A/UX use is assigned a slice number. Slice numbers are A/UX-specific: A/UX performs the service of translating slice numbers into the associated partition locations on the disk.



Name	Start Block	Length	Slice
MacPartition	16K	4000K	30
UNIX Root	4016K	83000K	0

Figure 4-2 Sample partition map

Identifying partitions

Slice numbers alone are insufficient to refer to partitions, since there can be partitions on various disks. For this reason, the SCSI ID number is also used, and both are made part of a filename construct that represents a logical disk device. Only selected administrative commands require you to refer to partitions by slice numbers, since this kind of access differs from the kind that is effected through the typical directory structure.

You can refer to a particular partition on particular disk drives according to the following pathname format:

```
/dev/dsk/cXdYsZ
```

The value of *x* is the SCSI ID of the hard disk; the value of *y* is the number of the subdrive at that SCSI ID; and the value of *z* is the slice number associated with a particular disk partition.

Some controller boards support multiple disk drives from one SCSI ID; in these cases the parameter *y* selects between those drives. One example is a Macintosh computer with multiple floppy disk drives, which uses the *y* parameter to distinguish the drives. However, all of the hard disks that you connect to the Macintosh through its built-in SCSI port have separate SCSI ID numbers; none will be identified as subdrives. Therefore, the value of *y* will be 0 for these drives.

For example, a partition on the internal hard disk of a Macintosh computer is identified by the device name `/dev/disk/c0d0sz`. A partition on the first external hard disk (with SCSI ID number 5) is identified as `/dev/disk/c5d0sz`. The following is a short list of conventions that A/UX currently employs:

- For the boot drive, slice 0 always refers to the root partition.
- For the boot drive, slice 1 refers to the swap partition.
- For the boot drive, slice 2 is reserved for the `/usr` file system, if it exists as a separate file system.
- Slices 25-30 are used for HFS partitions.
- Slice 30 always refers to the MacPartition (the Macintosh volume).
- Slice 31 always refers to the entire disk.

◆ **Note** The A/UX Startup application can directly access files in A/UX file systems. See the section “Accessing the A/UX File System From A/UX Startup” in Chapter 10. ◆

File systems

Each partition contains a file system to organize the disk area so that files can be accessed easily. Each A/UX **file system** is the complete set of data structures, commands, and subroutines used to manipulate data stored on a physical device. A/UX allows multiple file systems on a single disk (each in its own partition), and all such file systems on all disks can be simultaneously active. A/UX supports several types of file systems: Berkeley (UFS, the default file system), AT&T (SVFS) and network (NFS).

A file system resides on a partition and contains the data structures (directories and files, among others) that implement all or part of the A/UX directory hierarchy. The A/UX **directory hierarchy** is simply the collection of all files currently available to the system. From the user’s point of view, the directory hierarchy resembles an inverted tree, branching out from the root directory (`/`). Figure 4-3 shows two directory hierarchies, each in a separate partition.

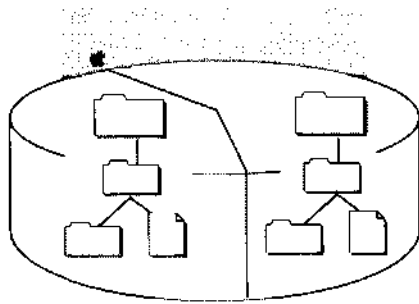


Figure 4-3 Directory hierarchies on separate partitions

An A/UX file system contains other internal organizational structures as well as files and directories. Among the structures defined within a file system are **superblocks**, which contain information about the file system, such as size, block size, number of inodes, and a pointer to a bitmap of free blocks. The default size for an A/UX file system block is 8 kilobytes and is usually the smallest unit of information passed to and from the hard disk. Each time a file system is made available to the operating system, the kernel sets aside a buffer in memory to hold its superblock. The kernel reads the superblock from disk into memory. Any changes made to the file system are first made to the image of the superblock in memory and periodically written back out to disk.

Accessing file systems

From the user's point of view, an A/UX file system is a hierarchical collection of directories and files seamlessly spreading from the root directory. To the system administrator, the hierarchical structure is maintained by attaching file systems to other file systems at directory locations called **mount points**. A mount point is a directory that serves as both a point of attachment and as a point of reference. The **mount** process is the means by which the partition-referencing slice numbers and the directory-based file references become connected. When a mounted file system appears in a file listing (see `ls(1)`), it seems like any other directory. A list of mounted file systems can be obtained with the `df` or `mount` commands.

Figure 4-4 illustrates the use of mount points, in this case `/pubs` and `/users`. The dashed lines represent the mounting process. Before file systems are mounted on them, the `/pubs` and `/users` directories would typically be empty. (However, if a mount point directory is not empty, the files it contains are inaccessible as long as the mount is in place.)

The `mount(1M)` and `df(1)` commands tell you which file systems are mounted and where they are mounted.

For example, if the `mount` command is issued for the system pictured in Figure 4-4, the result might look something like

```
/dev/dsk/c0d0s0 on /type 4.2 (rw,noquota)
/dev/dsk/c0d0s0 on /users type 4.2 (rw,noquota)
/dev/dsk/c5d0s3 on /pubs type 4.2 (rw,noquota)
```

The file called `text` in Figure 4-4 is known to be in the file system mounted at `/pubs` because of the hierarchical relationships that exist: `text` belongs to the directory `/sys`, which belongs to the mount-point directory `/pubs`. As you would expect in a hierarchical organization, everything located under `/pubs` is part of the file system symbolically known as `/pubs`. However, you cannot tell what disk and what partition `/pubs` corresponds to simply from knowing that `/pubs` is a mount point. You would need to run the `mount` or `df` command to get this information.

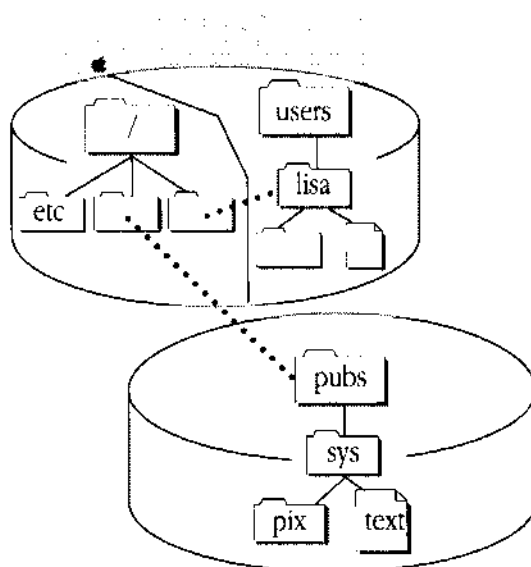


Figure 4-4 The mounting of file systems

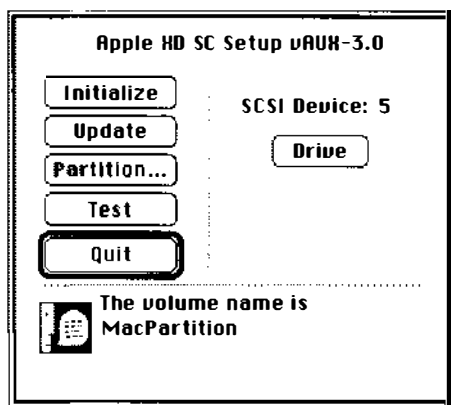
Technically, the mount point is the name of a directory. However, when a file system is customarily accessed through the mount point, this mount point is also used as the name identifying the file system. In the example above, you might make reference to the “pubs” file system and the “users” file system.

Because the directory hierarchy appears seamless, many users of A/UX won't care where the mount points are, particularly if they don't have to administer the system by mounting and unmounting file systems. The mounting and unmounting commands are tightly secured in most UNIX implementations, usually only the user `root` is allowed to perform these administrative tasks.

◆ **Note** A partition that does not contain a UNIX file system (such as the Swap partition) cannot be mounted. ◆

Background on Apple Hard Disk SC Setup

The Hard Disk SC Setup 3.0 program offers a variety of functions from its opening dialog box, including disk initialization and disk partitioning, as shown below.



◆ **Note** Hard Disk SC Setup 3.0 supports most third-party manufacturer's hard drives. There are some, however, with which it will not work. For this small set of drives, you must use the manufacturer's partitioning software or a non-Apple alternative. ◆

When you choose disk initialization, the system formats the disk and then tests it by writing various data patterns to all locations on the disk and verifying them.

▲ **Warning** Initialization erases all information previously on the disk. ▲

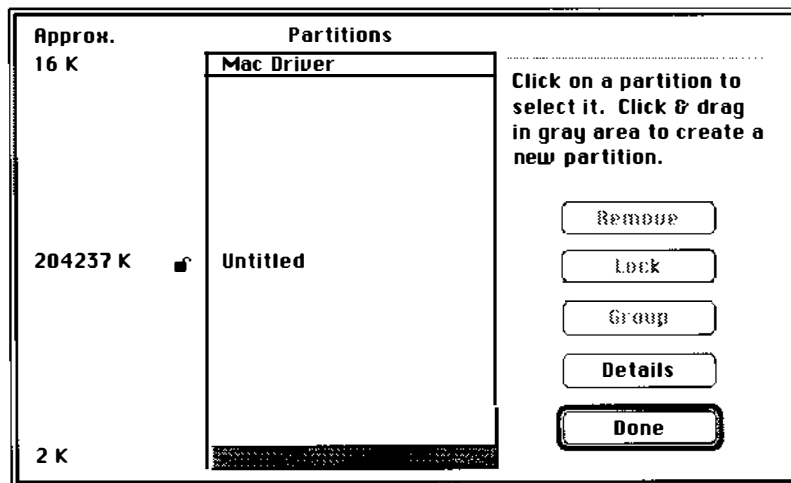
Additionally, the system builds a bad block table, flagging any defective areas found on the disk so that they won't be used.

After that, Hard Disk SC Setup builds a default partition map, indicating that one Macintosh partition of maximum size is desired. Those users that want to have UNIX partitions on the disk can repartition the disk easily by removing the Macintosh partition and adding UNIX partitions.

Next, Hard Disk SC Setup asks you to provide a name for this Macintosh **volume** (another term for a file system), which it has automatically created.

If you do not select the Partition button at this point and instead click Quit, this single default partition is written in the partition map on the disk (see the following dialog box). If you elect to make UNIX partitions on the disk, and are running Hard Disk SC Setup under A/UX, the system will make a default file system for you. When creating UNIX partitions that do not have a predefined mount point, such as / or /usr, you are prompted for a mount point (which need not exist).

During partitioning, you do not access the partitions themselves. Instead, you alter the partition map, which in turn changes the configuration of partitions.



Choosing a partition

For the administration of disks within A/UX, you must use the slice number and the SCSI ID number to refer to a partition on a particular disk, for example,

```
/dev/dsk/c5d0s3.
```

Hard Disk SC Setup automatically gives fixed slice numbers to partitions, as long as they are selected to be certain UNIX types of partitions. See “Adding a Partition,” later in this chapter for a list of these partitions and their given slice numbers.

Within Hard Disk SC Setup, referring to a partition is simple; you need only click a screen graphic that represents all of the partitions that exist. Accessible A/UX partitions have the slice number identified in their title. From a system viewpoint, Hard Disk SC Setup does not actually have to access the partition, because the program merely updates the partition map. However, in doing so, substantial or total data loss can result because the disk mapping has changed. The location of the physical data may be lost because the data location is found by adding the start of the partition to the offset. If either is unknown the data is inaccessible.

You can also alter the partition map with the `dp` utility. Use it with caution because it can disrupt the partition map—for instance, by creating overlapping, unreliable partitions. Probably the only reasons you would ever want to use `dp` is as a debugger, similar to `adb` or `fsdb`, or if you want to work with A/UX partitions of type Misc UNIX. See Appendix E, “Using the `dp` Utility,” for information about using `dp` to make changes to a partition map.

- ▲ **Warning** Hard Disk SC Setup is a powerful tool for reorganizing your hard disk. A complete, verified backup of the disk you intend to use Hard Disk SC Setup on should be made before launching the HD SC Setup icon. Repartitioning disks that already have files on them *will* result in significant or total data loss. ▲

When you alter the partition map by resizing or rearranging any of the partitions, the Macintosh and A/UX operating systems lose their ability to locate the start of the old partitions and any files within them. In effect, you lose all of the old files. This action will become clearer when you learn more about the details of A/UX disk access in the following section. It is for this reason you must completely back up any disk before using Hard Disk SC Setup on it.

Partitioning disks

Hard Disk SC Setup provides A/UX users with an easy way to allocate space for UNIX partitions. You can use Apple Hard Disk SC Setup 3.0 with most hard drives designed for use with a Macintosh computer.

See *Setting Up Accounts and Peripherals for A/UX* for step-by-step instructions for partitioning a disk for

- UNIX user files only
- a `/usr` partition and UNIX user files
- UNIX user files and the Macintosh OS

This chapter gives instructions for adding, removing, and moving partitions. It also provides instructions for grouping free space and increasing the size of your Swap partition.

If there is no free space in which to create new partitions, you must remove a partition and then add partitions in this newly gained space. To resize an existing partition, you must remove it, then recreate it at the desired size (see the following sections, “Removing a Partition” and “Adding a Partition”).

Considerations before you begin

For detailed hardware installation instructions, refer to the manuals provided with your hard disk drive. The most important concerns regarding the hardware setup are the following:

- Make sure the power is off to both your system and the drive while you connect the hardware.
- Set the SCSI ID number so that it doesn't conflict with existing SCSI devices. Refer to the documentation that came with your drive for instructions on how to set the ID number.
- Terminate only the first and last physically connected SCSI devices in the chain. Internal disk drives are usually already terminated. If you do not have an internal disk, you must put a terminator on the first (and last) external SCSI device. Refer to the documentation that came with your drive for termination instructions.

- Important** Hard Disk SC Setup 3.0 runs under both the Macintosh OS and the A/UX operating system. In order to take full advantage of increased A/UX functionality, it is recommended that you run it under A/UX.

Removing a partition

To make space for new custom partitions, you can remove existing ones. Working in the Custom Partition dialog box, you can remove any partition from your hard disk.

◆ **Note** In order to remove a partition containing a Macintosh volume, file sharing must be turned off. File sharing locks all Macintosh volumes while it is on. ◆

The following steps lead you through the removal of a partition.

1 Log in to A/UX as root.

2 Backup all files on your A/UX disk.

Before making any change to disk partitions, be sure you have reliable backups of everything on the disk.

3 Launch Hard Disk SC Setup.

Double-click the Apple HD SC Setup icon in the `/mac/bin` folder.

4 Select the desired drive.

Click the Drive button until you have selected the drive from which you want to remove a partition.

5 Select the partitioning function.

Click the Partition button.

6 **Select custom partitioning.**

Click the Custom button.

^ **Important** Don't remove the Mac Driver partition unless you have a special reason. Without the driver, you won't be able to use your disk after restarting. /

7 **Select the partition you wish to remove by clicking anywhere in its rectangle.**

The name of that partition is highlighted to show that it has been selected.

▲ **Warning** Removing partitions destroys any data previously in the partition. ▲

8 **Click Remove.**

An alert dialog box asks you to confirm that you want to erase the information in the partition.



9 **Click OK.**

Click Cancel if you decide not to remove the partition.

When you remove a partition, the representation of the space it occupied becomes gray to represent free space. If another area of free space is adjacent, the two rectangles are combined. If you want to combine two non-adjacent free spaces, you must group them, see the section "Grouping Free Space," later in this chapter.

10 Click Done in the Custom Partition dialog box to return to the main Hard Disk SC Setup dialog box.

When you are satisfied with the partitioning scheme you have obtained using the subprocedures, click Done.

11 Click Quit to return to the desktop.

12 Log out.

Adding a partition

Before you can add a new partition, there must be a section of free space large enough to hold it. If there is insufficient free space, remove one or more partitions. (For more information, see the preceding section, “Removing a Partition.”)

If the free space is divided into sections by existing partitions, with no single section large enough to hold your new partition, you need to remove or move a partition, or combine the sections of free space by grouping. (For more information, see the next section, “Grouping Free Space.”)

The following steps lead you through the addition of a new partition.

1 Log in to A/UX as root.

2 Backup all files on your A/UX disk.

Before making any change to disk partitions, be sure you have reliable backups of everything on the disk.

3 Launch Hard Disk SC Setup.

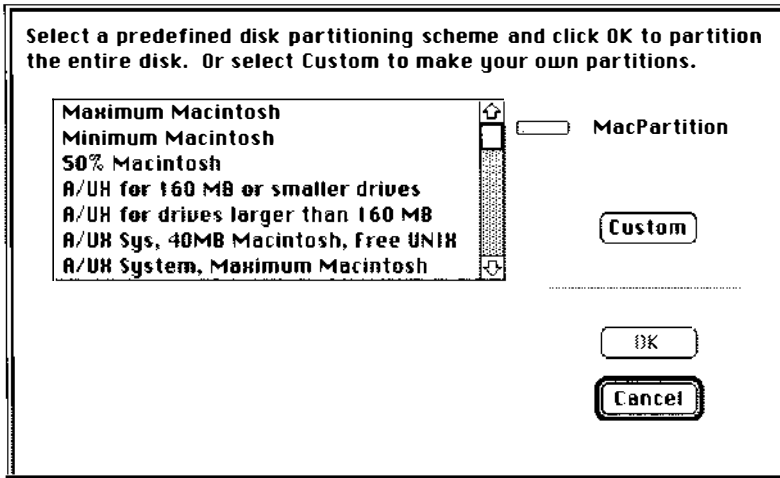
Double-click the Apple HD SC Setup icon in the `/mac/bin` folder.

4 Select the desired drive.

Click the Drive button until you have selected the drive on which you want to add a partition.

5 Select the partitioning function.

Click the Partition button. The predefined partitions available are displayed in the following dialog box.



6 Select Custom partitioning.

Click the Custom button.

7 Move the pointer to a gray rectangle representing free space and click in that space.

Hard Disk SC Setup draws two brackets representing the new partition. If you place the pointer in the upper half of the free space rectangle, the brackets start at the top of the free space; if you place the pointer in the lower half of the free space rectangle, the brackets start at the bottom of the free space. If no grey space is available, you cannot add a partition without removing one first.

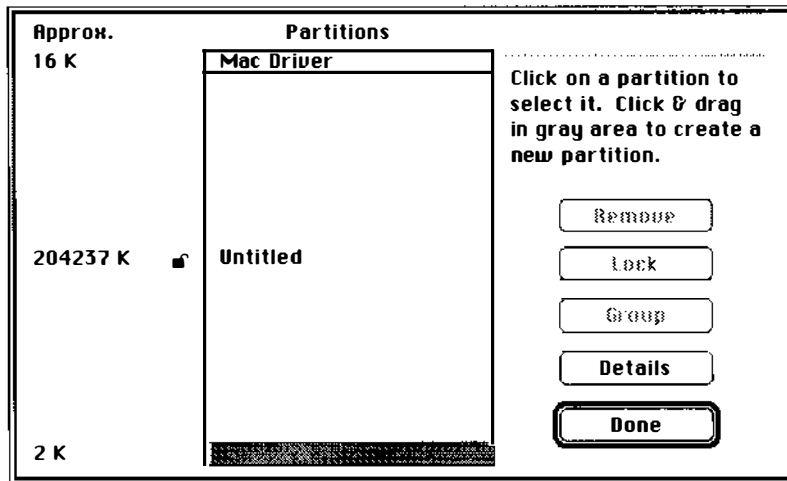
A shortcut to bypass steps 8 through 11 is to click in the top of the gray area, which presents the Partition Type dialog box, as shown in step 10 below. Enter the amount in the Adjust Size box and go to step 12.

Important You should leave at least 1K of free disk space at the bottom of the partition display. This is required to be able to test the disk.

8 Drag the pointer up or down to adjust the size of the new partition.

If you move the pointer to the left or the right of the rectangle, the brackets disappear.

The size, in kilobytes, is shown on the left.



Important You should leave at least 1K of free disk space at the bottom of the partition display. This is required to be able to test the disk.

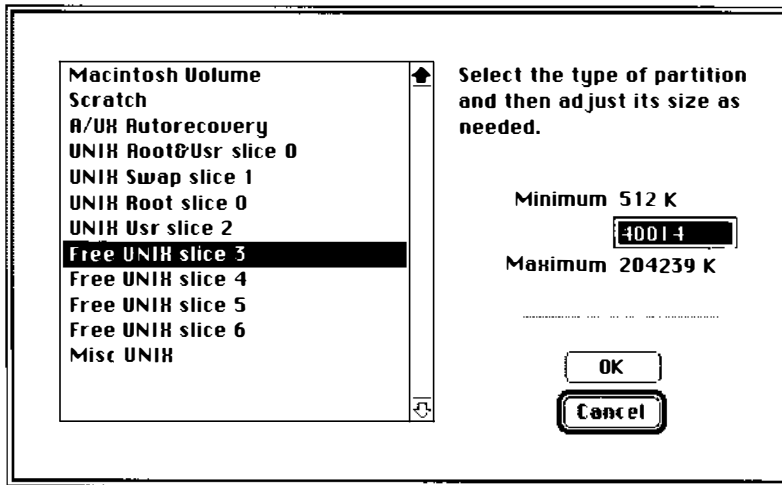
9 Release the mouse button when you are satisfied with the size of the new partition.

You don't need to be exact. You still have a chance to change the size.

Apple Hard Disk SC Setup immediately presents the Partition Type dialog box. You use the left side to select a partition type for the custom partition you are creating; you use the right side to adjust its size.

10 Select the partition type by clicking an item in the list on the left.

You have several choices for A/UX partitions, some of which are automatically assigned a slice number for use under A/UX.



Note that two of the A/UX partitions are not automatically assigned a slice number. They are A/UX Autorecovery and Misc UNIX. To associate these partitions with slice numbers, you must use the UNIX `dp(1M)` utility (see Appendix E, “Using the `dp` Utility”). Generally, the partition type Misc UNIX is best avoided, simply because of the inconvenience of using `dp`. Do not use the Autorecovery partition for personal files. You should never need to associate the Autorecovery partition with a slice because the autorecovery utilities will do this when necessary.

11 Enter the size in kilobytes, if you wish to change the size of the partition.

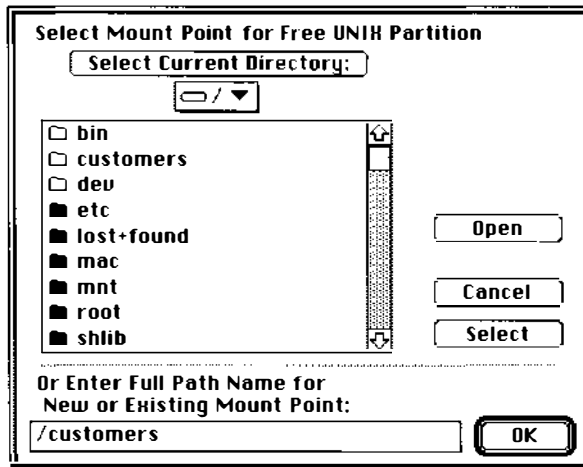
You can enter the partition size to a precision of a half-kilobyte (0.5K). Hard Disk SC Setup does not allow other fractions.

You can change the size by entering a new number for the size of the partition.

The maximum possible size is shown below the highlighted box; if you have selected a partition type, the minimum possible size is also shown.

12 Click OK.

The Hard Disk SC Setup program creates the new partition and initializes the file system (if appropriate). If you selected Free UNIX slice 3, 4, 5, or 6 you will then see the following dialog box, which prompts you for the name of a mount point. If you selected Macintosh Volume, the volume is given the name Untitled.



13 Select a mount point.

Click OK to select the default mount point name, or type a different mount point name in the box at the bottom of the dialog box, then click OK. The system displays a confirmation dialog box.



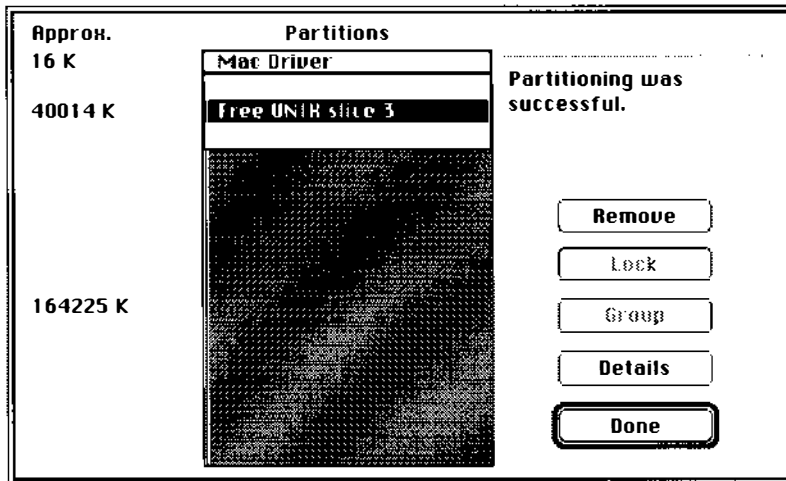
14 Confirm the mount point.

Click OK to confirm the mount point name.

You can create another custom partition as long as you have sufficient free space, but you can select only a listed partition type.

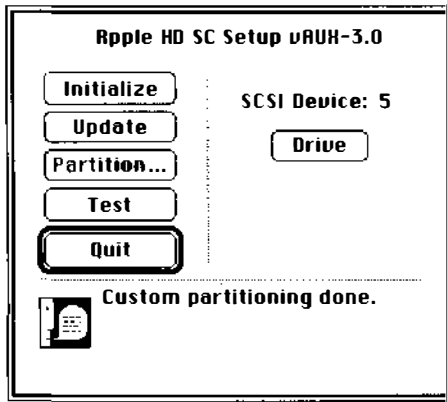
Click Cancel in the Partition Type dialog box if you wish to return to the Custom Partition dialog box without creating a new partition.

The Hard Disk SC Setup program creates the new partition, initializes the file system (if appropriate), confirms that the partitioning was successful, and again presents the Custom Partition dialog box, in which the new partition is shown.



15 Click Done in the Custom Partition dialog box to return to the main Hard Disk SC Setup dialog box.

When you are satisfied with the partitioning scheme you have obtained using the subprocedures, click Done. The program confirms that the custom partitioning is completed.



- 16 **Click Quit to return to the desktop.**
- 17 **Log out.**

Grouping free space

If you have created two or more partitions on your disk, and free space separates them, you may eventually end up with multiple free space areas that could be consolidated into one large free space area. Grouping combines the free space on your disk.

Sometimes this function is not available; for instance, when there are less than two free space areas remaining. At such times the Group button is dimmed.

- 1 **Log in to A/UX as root.**
- 2 **Backup all files on your A/UX disk.**

Before making any change to disk partitions, be sure you have reliable backups of everything on the disk.

- 3 **Launch Hard Disk SC Setup.**

Double-click the Apple HD SC Setup icon in the `/mac/bin` folder.

4 Select the desired drive.

Click the Drive button until you have selected the drive on which you want to group the free space.

5 Select the partitioning function.

Click the Partition button.

6 Select custom partitioning.

Click the Custom button.

7 Click Group.

Hard Disk SC Setup presents an alert box, warning that moving information from one portion of your disk to another will take time. Because grouping usually means that a large amount of information is being moved, Hard Disk SC Setup also warns that some information might be lost in this process.

8 Click OK.

All the free space on the disk is grouped together, and it is shown together at the bottom of the Custom Partition display.

Click Cancel if you decide not to group the free space.

9 Click Done in the Custom Partition dialog box.

10 Click Quit to return to the desktop.

11 Log out.

Moving a partition

You can also use the mouse to move a partition into adjacent free space or into any free space larger than the partition.

1 Login in to A/UX as root.

2 Backup all files on your A/UX disk.

Before making any change to disk partitions, be sure you have reliable backups of everything on the disk.

3 Launch Hard Disk SC Setup.

Double-click the Apple HD SC Setup icon in the `/mac/bin` folder.

4 Select the desired drive.

Click the Drive button until you have selected the drive on which you want to move a partition.

5 Select the partitioning function.

Click the Partition button.

6 Select custom partitioning.

Click the Custom button.

7 Click the partition to be moved.

4 **Select the desired drive.**

Click the Drive button until you have selected the drive on which you want to see the partition details.

5 **Select the partitioning function.**

Click the Partition button.

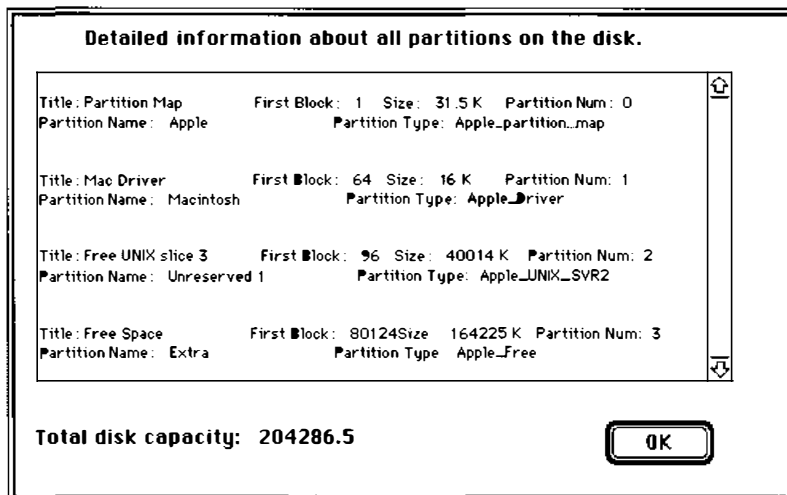
6 **Select custom partitioning.**

Click the Custom button.

7 **Click the Details button.**

The Details window, shown below, appears. It shows each partition, its name, type, size in kilobytes, and the block where the partition begins on the disk. At the bottom of the window, the total disk capacity is displayed.

The Details window shows one more type of partition, the Partition Map, which contains information about the partitions on the disk. You cannot directly change the Partition Map.



If you select a partition before clicking Details, that partition is highlighted in the Details window. If you select a partition in the Details window by clicking one of its lines, the partition is highlighted when you return to the Custom Partition dialog box. You can't select the partition map.

- 8 **Click OK to close the Details window and return to the Custom Partition dialog box.**
- 9 **Click Done in the Custom Partition dialog box.**
- 10 **Click Quit to return to the desktop.**
- 11 **Log out.**

Adding swap space

You can use the space allocated for a partition not only to create file systems but also to increase the UNIX swap area from the default 18 megabytes provided. There is one Swap partition on your startup drive; you can use Swap partitions on other drives as well. (See the section “Adding a Partition,” earlier in this chapter, for general instruction on adding partitions, including Swap partitions.)

Partitions that the system uses for swap space are associated with slice 1—the type Swap from Hard Disk SC Setup. Typically, you only need to add swap space to optimize performance; for example, the swap space should be between two and three times the size of physical memory. To add swap space from a Swap partition on a drive other than your boot drive, use the following procedure:

- 1 **Log in to A/UX as root.**
- 2 **Verify the size of the current swap space.**

Enter the following command:

```
swap -l
```

Note the size of the current swap area, which is listed in the `blocks` column.

3 Identify the additional swap space to the operating system.

Add the following line to the file `/etc/rc`:

```
swap -a /dev/dsk/cxxd0s1
```

where *x* is the SCSI ID number of the disk the additional Swap partition is on.

4 Log out and restart A/UX.

5 Log back in to A/UX as root.

6 Verify the size of the current swap space.

Enter the following command:

```
swap -l
```

The size listed in the `blocks` column should be larger than the size obtained in step 3.

To increase the size of a currently existing Swap partition on your boot drive, use the following procedure:

1 Log in to A/UX as root.

2 Backup all files on your A/UX startup drive.

Before making any change to disk partitions, be sure you have reliable backups of everything on the disk.

3 Log out of A/UX and restart your system.

4 Launch Hard Disk SC Setup from the Macintosh OS.

Restart the system. Once the Finder has appeared, insert the Hard Disk SC Setup disk. Double-click the Apple HD SC Setup icon.

5 Select your A/UX boot drive.

Click the Drive button until you have selected your A/UX boot drive.

6 Select the partitioning function.

Click the Partition button.

7 Select custom partitioning.

Click the Custom button.

8 Remove partitions, if necessary, to make room for the increased swap space.

Click the partition to be removed, then click the Remove button. Click the OK button.

▲ **Warning** Removing partitions destroys any data previously in the partition. ▲

9 Remove the A/UX Swap partition.

Click the A/UX Swap region. Click the Remove button. Click the OK button.

10 Group the free space, if necessary.

Click the Group button.

11 Recreate the A/UX Swap partition using the free space on the disk.

Click the gray region representing free space on the disk. Click the UNIX Swap slice 1 choice from the list at the left side of the dialog box. Enter the desired swap space size in the highlighted window. Click the OK button.

12 Exit Hard Disk SC Setup.

Click the Done button. Click the Quit button.

13 Restart A/UX.

Your system will have access to the additional swap space.

Making A/UX file systems

A/UX Release 3.0 supports both Berkeley (UFS) and System V (SVFS) file systems. The Berkeley file system is created by default in A/UX, as it has several advantages over the System V file system:

- UFS increases file system reliability by creating multiple copies of the superblock and positioning them around the disk. This increases the probability that the file system can be repaired in the event of a media failure. SVFS uses only one copy of the superblock.
- UFS increases disk storage efficiency by utilizing data block fragments to store portions of a file that do not fill an entire data block.
- UFS increases disk performance by using several local inode lists that are positioned near the inodes to which they refer. SVFS uses one inode list and must do a relatively long disk seek to get access to the inodes to which it refers.

If you would prefer to use Commando dialog boxes instead of the command line interface to make file systems, see *Setting Up Accounts and Peripherals for A/UX*.

Disk parameters: `/etc/disktab`

In order to optimize the access speed of disks, the file system contains information on the geometry of various types of disk drives. For SVFS file systems, a calculation based on the geometry information must be entered on the command line when the file system is made. For UFS file systems, this information is stored in the file `/etc/disktab`. While there is a representative sampling of Winchester-technology disk drives included in the file, not all drives are listed. You can add entries for your particular drive in this file to maximize the performance of your system.

A sample entry in the `disktab` file is:

```
# HD 80 SC (80 MB Quantum Q280)
#
Q280|q280|HD80SC:\
      :ty=winchester:ns#32:nt#6:nc#823:
```

The first uncommented line gives the names by which disks with these parameters are known. Note that the names are separated by a `|` character. The second line gives various parameters of the drive; the parameters are separated by colons (`:`). Parameters for which you typically supply values are

- `ty` The kind of disk drive. In almost all cases on A/UX systems, this parameter is `winchester`.
- `ns` The number of sectors per track.
- `nt` The number of tracks per cylinder. This is equal to the number of heads on the disk.
- `nc` The total number of cylinders on the disk, minus three. Three cylinders are reserved for drive testing.

Values for the `ns`, `nt`, and `nc` parameters are available from the drive manufacturer. Also, some disk partitioning software can read this information directly from a disk.

Making Berkeley file systems: `newfs`

The `newfs` program creates file systems by placing the correct initial values into a superblock and storing them at the starting block offset for the associated partition.

- 1 Log in to A/UX as root.**
- 2 Examine the file `/etc/disktab` to determine the name of the disk you have.**
See the section “Disk Parameters: `/etc/disktab`,” earlier in this chapter.
- 3 Make a file system inside an existing partition.**

Enter the command:

```
newfs /dev/dsk/cxd0sz name
```

where

x is the SCSI ID number of the disk.

z is the disk slice number.

name is the name of disk as specified in `/etc/disktab`.

The system will respond with a list of the backup superblock locations.

4 **Check the consistency of the new file system.**

Enter the command:

```
fsck -y /dev/dsk/cxd0sz
```

where *x* and *z* are the values used in step 3.

For example, if the SCSI ID number of your new external drive is 6 and the slice number of the partition is 0, enter the command

```
fsck -y /dev/dsk/c6d0s0
```

If any inconsistencies exist, `fsck` automatically repairs them for you.

◆ **Note** The file system initially created by `newfs` contains one directory called `lost+found`, which is where the file system check program (`fsck`) stores files and directories that have become disconnected from the file system. ◆

5 **Complete the procedure in “Mounting A/UX File Systems,” later in this chapter, to make the disk accessible as part of the A/UX file hierarchy.**

Making System V file systems: `mkfs`

Making a System V file system is slightly more complex than making a Berkeley file system because you must calculate the disk size and enter the size on the command line.

1 **Log in to A/UX as root.**

2 Make a file system inside an existing partition.

Enter the command:

```
mkfs /dev/dsk/c $x$ d0s $z$  size
```

where

x is the SCSI ID number of the disk.

z is the A/UX disk slice number.

size is the number of 512-byte blocks in the partition. It is calculated as $(2 * \text{size in kilobytes})$.

3 Check the consistency of the new file system.

Enter the command:

```
fsck -y /dev/dsk/c $x$ d0s $z$ 
```

where x and z are the values used in step 2.

For example, if the SCSI ID number of your new external drive is 6 and the slice number of the partition is 0, enter the command

```
fsck -y /dev/dsk/c6d0s0
```

If any inconsistencies exist, `fsck` automatically repairs them for you.

4 Complete one of the procedures in the next section, “Mounting A/UX File Systems,” to make the disk accessible as part of the A/UX file hierarchy.

Mounting A/UX file systems

Once a file system has been made, it must be mounted to make it accessible through the normal A/UX directory structure. You can have the system mount the files automatically when the system starts by using the `fentry` command to make an entry in the `/etc/fstab` file. Alternately, you can mount file systems temporarily, unmounting and remounting them as needed.

There are two methods to mount file systems (described in the following sections):

- Automatically, using the `fentry` command.
- Step-by-step, using the `mount` command.

Both methods can be completed by using the command-line interface. If you would prefer to use Commando dialog boxes instead of the command line interface to make file systems, see *Setting Up Accounts and Peripherals for A/UX*.

Mounting automatically: `fentry`

The `fentry(1M)` command creates an entry in the file system table, `/etc/fstab`, and automatically mounts a file system. (You can set an option in the command line to override the automatic mounting action.) After the entry is made, the file system will automatically be mounted when the system starts. You can make one entry each time you invoke the `fentry` command. For a description of the Commando dialog box for `fentry`, see *Setting Up Accounts and Peripherals for A/UX*.

To mount a file system automatically, follow this procedure:

- 1 Log in as root.**
- 2 Run the `fentry` command.**

Enter the command:

```
fentry -p 2 -t type block-device mount-point
```

where the command line arguments are:

- | | |
|----------------------------------|---|
| <code>-t <i>type</i></code> | The type of file system: enter <code>4.2</code> for UFS; <code>5.2</code> for SVFS (or <code>nfs</code> for NFS). |
| <code><i>block-device</i></code> | The file system to be mounted; for example, <code>/dev/dsk/cxd0sz</code> , where <code>x</code> is the SCSI ID number of the hard disk that contains the file system, and <code>z</code> is the slice number (usually 0 to 6, inclusive—never use slice 30 or 31). |
| <code><i>mount-point</i></code> | The full pathname of a directory on the local machine that is to be used as a mount point. The <code>fentry</code> command creates this directory if it does not exist. For example, an A/UX user file system may be mounted at <code>/users</code> (the A/UX convention) or <code>/user</code> . |

For example,

```
fscopy -p 2 -t 4.2 /dev/dsk/c5d0s3 /user
```

creates a file system table entry that automatically mounts the UFS file system located at slice 3 of the SCSI drive at address 5 (`/dev/dsk/c5d0s3`) to the mount point directory `/user`.

3 Log out.

◆ **Note** If you have more than one A/UX file system on the disk, use the `fscopy` command once for each of them. ◆

See the `fscopy(1M)` man page in *A/UX Command Reference* for additional options that allow you to override default values.

Mounting temporarily: `mount`

You can mount file systems for the duration of a login session, or until you unmount them, by using the `mount` command.

To mount a file system temporarily, follow this procedure:

1 Log in as root.

2 Create a mount point, if necessary, for the new file system.

You can use any directory for a mount point. It is usually best to use an empty directory because the mounted file system overlays any files in the directory, making them inaccessible for the duration of the temporary mount. For example, the empty directory `/mnt` is shipped (or installed) on your A/UX startup disk as a convenient mount point. Reserve `/mnt` for *temporary* file systems; for example, you may wish to use `/mnt` for mounting file systems on removable floppy disks.

To create a directory intended for use as a mount point, enter

```
mkdir mount-point
```

For the example in the next few steps, slice 0 of your external drive with SCSI ID number 6 is assumed to be dedicated to hold documentation. In that case, the following command line makes a descriptive mount point:

```
mkdir /pubs
```

The directory `/pubs` is the mount point where you could attach the newly created file system.

3 **Set the permissions for the mount point directory.**

Enter the command

```
chmod 444 mount-point
```

This setting prevents users from accidentally creating files in the mount point directory when no file system is mounted.

4 **Mount the new file system.**

Enter the command

```
mount -v /dev/dsk/cxxd0s $z$  mount-point
```

where

x is the SCSI ID number of the disk.

z is the A/UX disk slice number.

mount-point is the directory to which this file system is being mounted.

The command returns a message informing you that your new file system is now mounted at *mount-point*.

For example, to mount a floppy disk (which must already have an A/UX file system made on it, and be inserted in drive 0 of the Macintosh computer) at the directory `/pubs`, use the command:

```
mount -v /dev/dsk/c8d0s0 /pubs
```

To mount slice 3 of a hard disk (having SCSI ID 6) at the directory `/pubs`, use the command:

```
mount -v /dev/dsk/c6d0s3 /pubs
```

◆ **Note** The `mount` and `umount` commands refer to the file system type in `/etc/fstab`, or make use of the `-T` (type) option. ◆

5 Set the permissions for the mounted file system.

By setting the permissions on the mounted file system differently than those set for the mount point directory itself, users have an indication whether the file system is mounted or not.

For example, to give everyone access to the mounted file system's top directory, enter the command

```
chmod 777 mount-point
```

6 Verify the file system is mounted.

To verify that your new file system is mounted, enter the command:

```
mount
```

Your system will respond with a list of the currently mounted file systems and their mount points.

7 Log out.

◆ **Tip** To preserve a list of the currently mounted file systems for later automatic mounting, log in as `root` and enter the following commands:

```
cp /etc/fstab /etc/fstab.old  
mount -p > /etc/fstab
```

This overwrites the `fstab` file, but saves you a lot of time if you want to preserve the current mount state. ◆

Unmounting file systems: `umount`

To disassociate a file system from its mount point, use the `umount(1M)` command; follow this procedure:

1 **Log in as root.**

2 **Unmount the file system.**

Enter the command

```
umount mount-point
```

where *mount-point* is the directory to which this file system was mounted.

◆ **Note** The spelling of this command is `umount` *not* `unmount`. ◆

3 **Log out.**

If you attempt to unmount a file system and receive the error message

```
source: Device busy
```

this means that you or other users are accessing files or directories on the file system mounted on *source*. If you are in single-user state or are the only user on the system, simply change your current directory to a directory that is not on this file system and reenter the `umount` command. If other users are accessing this file system, ask them to finish their work and change their current directory to another so that you can unmount the file system.

▲ **Warning** Ejecting a mounted floppy disk before issuing the `umount` command can damage the file system because A/UX may still have file system data in a memory buffer. ▲

5 Backing Up Your System

Backup overview / 5-3

Backup media / 5-7

The backup utilities / 5-10

Verifying data on backed-up disks / 5-43

Automating system administration with `cron` / 5-45

This chapter discusses the various methods by which you can back up your system.

Backing up means copying the data on your hard disk to an alternative medium, such as a floppy disk or a magnetic tape, from which you can restore the data to your hard disk, if necessary.

Making regular backup copies of files and file systems is one of the most important duties of the system administrator. Computer data stored on disk can be damaged by hardware failure, or users may accidentally remove it. If you make regular backups, you have a better chance of being able to restore data that is damaged, lost, or destroyed.

Store the media on which your systems are backed up in a safe place—off site if necessary. Also, keep a backup log as a written record of what was backed up and when it was backed up.

There are many ways to back up data. To decide on the best technique, compare the time it takes to complete a backup with the time it may take to restore a backup. Also consider how often your data is changed, how valuable the data is, and how many people use the system. The safest plan is to devise an overlapping strategy, combining two or three backup techniques. Generally, if you use a regular schedule for full backups and supplement those with one or two partial backups, you can be assured that you can rebuild your system with minimal data loss. You may want to customize backup commands in a shell script to keep all backups consistent.

You may also use the A/UX backup utilities to store directories no longer needed on the system. By storing unused data on floppy disks or tapes, you free the system disk for use and improve performance.

The backup and restoration utilities available on the A/UX system include `cpio`, `tar`, `dump.bsd`, `restore`, and `pax`. A number of other companies also offer backup utilities; however, unless they have been certified to work with A/UX 3.0 they may not restore permission and owner information. Before using one of these utilities, confirm with its manufacturer that it preserves the UNIX permissions and ownerships of A/UX files.

Backing up and restoring are mutually dependent activities with `cpio` or `tar`. If you back up with `tar`, you can use only `tar` to retrieve the data; the same is true for `cpio`. The `pax` command, however, enables you to read or write `tar` or `cpio` archives. **Archives** are another term for copies of files or file system data in a particular format that the user stores on a removable medium, usually floppy disks or magnetic tape.

Backup overview

The following sections provide background information to familiarize you with the backup process.

Full versus partial backups

There are two main strategies for creating backups. The first is a **full backup**, during which all the data on each file system is copied; this is a time-consuming process. Full backups copy a system in such a way that you can reload it if your disk is completely erased. The second strategy is to perform a **partial backup**, during which only part of the system is backed up. This action is less time consuming and more useful for most types of everyday work. Partial backups can be either selective or incremental. To make a selective backup, you specify the files and directories according to your needs, such as specific filenames, or by user or group ownership. Generally, you use `tar`, `cpio`, or `pax` to back up specific data.

To make an incremental backup, the system uses the modification dates on files to automatically copy all newly created files and all files modified since the date of the last backup. You can use `cpio`, `dump.bsd`, and `pax` utilities to make incremental backups.

You can use the `find` command with the `-mtime` option to discover when a file was last modified, or the `-ctime` option to find out when it was last changed.

A common backup scheme

A commonly used backup scheme is illustrated in Figure 5-1, which shows four levels of backup. This arrangement ensures that your data is adequately backed up at all times.

You can recycle the tapes or disks used for backups once they are no longer useful. Typically, a backup is considered useful for two units of time beyond its creation date. For example, Monday's daily backup disk can be recycled Wednesday evening, keeping Tuesday's backup in reserve in case Wednesday's is damaged in some way. Remember

that floppy disks, cartridge tapes, streaming tapes, and certain other backup media can be used only a finite number of times; check the manufacturer's specifications to determine how many times you can safely recycle your backup media.

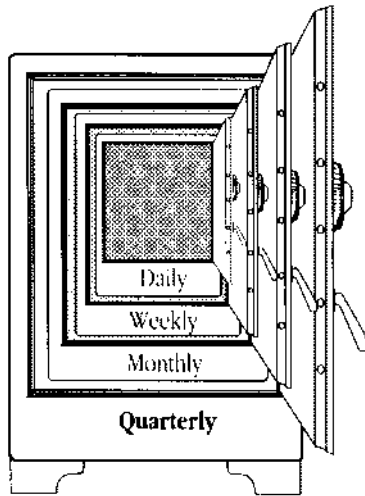


Figure 5-1 A common backup scheme

Referring to devices by device file names

When you use most backup utilities, you need to specify where the files you want to back up are and where the backups should be recorded. In A/UX, all peripherals (hard disk drives, floppy disk drives, tape units, terminals, and so on) are known to the system through **device files** located in the `/dev` directory. A particular peripheral device can be referred to by more than one device filename, which may result in different behavior, as shown in Table 5-1. When a backup utility reads or writes to such a file, that information is read (or written) to the peripheral corresponding to the device file.

There are two varieties of device file: **block devices** and **character devices** (also known as **raw devices**). The operating system transfers data to block devices after buffering it, while data is transferred in single bytes to character devices. Whether you access the device as a block device or as a raw device depends on the requirements of the utility you are using. Table 5-1 lists the names of the device files that are standard for A/UX.

Table 5-1 Standard A/UX device files

Device/File name	Peripheral Type
/dev/floppy0	floppy disk drive #0 block
/dev/rfloppy0	floppy disk drive #0 char
/dev/floppy1	floppy disk drive #1 block
/dev/fd/d0	floppy disk drive #0 block
/dev/fd/d0e	floppy disk drive #0, eject when done block
/dev/rfd/d1	floppy disk drive #1 char
/dev/rfd/d1ew	floppy disk drive #1, wait for disk, eject when done char
/dev/dsk/c8d0s0	floppy disk drive #0 block
/dev/rdsk/c8d0s0	floppy disk drive #0 char
/dev/dsk/cx \bar{d} y \bar{s} z	hard disk SCSI ID x , drive y , slice z block
/dev/rdsk/cx \bar{d} y \bar{s} z	hard disk SCSI ID x , drive y , slice z char
/dev/rmt/tc \bar{x}	tape drive SCSI ID x char
/dev/rmt/tc \bar{x} n	tape drive SCSI ID x , no-rewind char
/dev/rmt/tc \bar{x} .1600	tape drive SCSI ID x , 1600 bpi density char
/dev/rmt/tc \bar{x} .6250	tape drive SCSI ID x , 6250 bpi density char

Important A user (especially root) should never attempt to write data directly using the name of a device file that accesses a disk or tape drive. Not using the appropriate utilities can result in complete data corruption. The tape device filenames should be used only on backup utility command lines.

As shown in Table 5-1, you can use `/dev/dsk/c8d0s0` to refer to floppy disk drive #0. This nomenclature reflects the SCSI naming scheme, first mentioned in Chapter 4. Devices in the `/dev/dsk` and `/dev/rdsk` directories are named according to their controller, drive, and partition number in the following format:

`/dev/dsk/cx \bar{d} y \bar{s} z`

The value of x is the SCSI ID of the hard disk; the value of y is the number of the subdrive at that SCSI ID; and the value of z is the slice number associated with a particular disk partition. Subdrives are usually used only for floppy disk drives.

For example, `/dev/dsk/c0d0s0` signifies SCSI ID 0, the first drive, and the first assigned UNIX partition. The same partition on an external hard disk with SCSI ID 5 would be referred to as `/dev/dsk/c5d0s0`. See the section “Disk Partitions” in Chapter 4 for more information on the SCSI naming scheme and a discussion of partition slices.

Tape devices may have several device names that refer to different capabilities of a single piece of hardware. For example, the tape devices `/dev/rmt/tcx.1600`, `/dev/rmt/tcx.3200`, and `/dev/rmt/tcx.6250`, allow you to choose various recording densities to use with 9-track tape drives.

Floppy devices can also have several names referring to different capabilities. For example, the disk device `/dev/fd/d0e` ejects a disk after it has finished writing or reading the disk. The device `/dev/fd/d0w` waits for a disk to be inserted before writing or reading the disk. The disk device `/dev/fd/d0` automatically determines the density of a disk inserted into drive 0, while the device `/dev/fd/d0m800` ejects any disk that is not specifically an 800K disk.

Multiple archives on a single tape

A/UX 3.0 provides support for **no-rewind devices**, which allow you to put several backups on a single tape. A no-rewind device simply doesn't rewind the tape after it finishes writing (or reading) a backup. The next backup can then be written (or read) from that point in the tape. With the advent of large tape-storage capacity, this can be helpful in keeping down the costs of preserving your data. The `mt(1)` command makes it easy to work with multiple tape archives. This command lets you move forward or backward between archives, format tapes (if necessary), and fully rewind a tape. The regular rewind devices have names like `/dev/rmt/tcx`, where *x* is the SCSI address of the tape drive. You specify the no-rewind tape device by adding an `n` after the device name: `/dev/rmt/tcxn`. See the section “Creating Multiple-Archive Backups,” later in this chapter, for an example of using no-rewind devices.

Run levels and backups

Generally it is best to make backups of file systems that are not being used. If you are not on a network and not running any background processes, any time is a good time to do backups. If you are on a network, however, you may not always know when another user starts changing things in a file system you may want to back up. One of the simplest ways to stop network access is to take your system to single-user state.

In the process of getting to single-user state, all file systems are unmounted. In order to use the `cpio`, `tar`, or `pax` utilities, you will need to temporarily mount the file system you want to back up. (See the section “Mounting A/UX File Systems” in Chapter 4 for a description of the mounting process.) The `dump.bsd` utility can back up unmounted file systems.

Backing up mounted file systems

When file systems are mounted you can traverse the directory tree of the file systems as if they were ordinary branches of the root file system. This means backup utilities that use the regular directory structure cannot distinguish between one file system and the next. Consequently, an administrator backing up data must be careful when deciding the directory in which to start the backup. This is particularly true when network file systems (NFS) may be mounted. Since these file systems also appear to be part of the file hierarchy, a backup utility may inadvertently attempt to back up an astonishing amount of data. You can avoid this possibility by unmounting all NFS file systems before starting your backup. Another possibility is to unmount the file system you want to back up and use the `dump.bsd` utility on the unmounted file system.

Backup media

You can make backups on a variety of media. Currently, you can back up UNIX files onto floppy disks, hard disks, cartridge tapes, or nine-track magnetic tape.

Storage capacity of backup media

The capacity of storage media varies widely. Table 5-2 shows the capacities of various disks and tapes. This table also shows ranges for some capacities because different models of tape drives can record different amounts of information on the same length tape.

Table 5-2 Backup media capacities

Backup media	Type	Capacity
Floppy disk	single density	400 KB
	double density	800 KB
	high density	1.4 MB
Apple 40SC	cartridge tape	38.5 MB
8 mm DAT	15m cartridge tape	300 MB–600 MB
	54m cartridge tape	1.2 GB–2.5 GB
	112m cartridge tape	2.5 GB–5.0 GB
1/4" QIC	cartridge tape	150 MB–525 MB
4 mm DAT	cartridge tape	2 GB
DCAS	cartridge tape	150–600 MB
9-track	2400 ft. tape	40–144 MB

When to use floppy disks

Use floppy disks to store backups when

- backing up a small amount of data, such as several data files
- making a daily backup; that is, backing up files changed that day

You can use the `cpio`, `tar`, `pax`, and `dump.bsd` utilities to make backups on floppy disks. This chapter describes how to use these A/UX utilities.

When to use tape

Tapes can hold between 38.5 megabytes and 5.0 gigabytes of data, depending on the technology used: 1/4 inch cartridges, 9-track reels, digital cassette (DCAS) cartridges, or digital audio tape (DAT) cartridges. Therefore, use tapes when you back up large amounts of data—for example, when you make a full backup. Use tapes when

- making weekly or monthly backups
- backing up large amounts of data, such as entire file systems

Table 5-3 lists various tape formats and the backup utilities compatible with them. This chapter describes later how to use these A/UX utilities to back up to tape.

Table 5-3 Backup utility compatibility

Backup media	Archival utilities	Copy utilities
Floppy disk	cpio, tar, dump.bsd, pax	
Apple 40SC	cpio, tar, dump.bsd,	volcopy, dd
8 mm DAT	cpio, tar, dump.bsd	volcopy, dd
1/4" QIC	cpio, tar, dump.bsd	volcopy, dd
4 mm DAT	cpio, tar, dump.bsd	volcopy, dd
DCAS	cpio, tar, dump.bsd	volcopy, dd
9-track	cpio, tar, dump.bsd, pax	volcopy, dd

See *Setting Up Accounts and Peripherals for A/UX* for instructions on formatting a tape in the Macintosh OS and on adding an Apple Tape Backup 40SC to or removing it from your A/UX system.

Though the Apple Tape Backup 40SC drive and cartridges are compatible with A/UX 3.0, the Apple Tape Backup 40SC software is incompatible with A/UX 3.0. Therefore, you can no longer use it to back up entire disk partitions. To back up entire disk partitions onto Tape Backup 40SC cartridges, use either `volcopy(1M)` or `dd(1)`. The other utilities available for use with these cartridges are listed in Table 5-3.

When to use paper

For certain system files it is handy to have paper records of the contents of the files, as a further protection against catastrophic loss that might include the loss of your backups. Treat paper records with the same security precautions as your backups, as they likely contain confidential information. Candidates for paper copies are:

```
/etc/fstab /etc/hosts  
/etc/group /etc/passwd
```

You may also want to have a printed report listing all the files in your system when you make a full backup. This helps in locating a particular file should you need to restore it. The commands

```
ls -Rli / > backuplist  
lpr backuplist
```

provide a printed list of important data about each file in your system such as the size, owner, and modification time. These commands create a copy of the file on your disk as well.

The backup utilities

Backup utilities fall into two categories, archival and copy utilities. This chapter describes **archival utilities**, which copy data and related reference information that records the position of the data within the file system. Archival utilities are designed specifically to move files and directories between media; the reference information, such as the pathnames to the current directory, helps to reconstruct the original data structure.

The **copy utilities** copy data only; they copy byte-for-byte and disregard any structure that the data may have. These utilities are useful if you are going to copy an entire file system to another disk. However, it is not very easy to recover individual files from these backups. For information on these programs, see `dd(1)` in *A/UX Command Reference* and `volcopy(1M)` in *A/UX System Administrator's Reference*.

Archival backup utilities include `cpio`, `tar`, `dump.bsd`, and `pax`. All of the backup utilities share these advantages:

- They allow you to back up and retrieve individual files, directories, and file systems, or any combination of the three.
- They are convenient for copying the entire contents of a directory tree.
- They give you the choice of making full backups (of entire file systems) or partial backups (of specific files).

About the Finder

You can back up A/UX files and Macintosh documents by dragging their icons onto Macintosh-formatted disks. This is by far the easiest way to back up files. However, A/UX files that are copied in this way do not retain their permissions or dates, so this method isn't recommended for files that will typically be used in the A/UX file system, such as programs and shell scripts. When these files are dragged back into the A/UX file hierarchy, they will be owned by the current logged-in user and given the permissions specified by the current `umask` value. See Chapter 3 for a discussion of the `umask` parameter. Dragging files created by Macintosh applications should not result in permission problems.

About `cpio`

The utility `cpio` backs up and restores an entire file system or individual files. The advantages of using `cpio` include:

- You can restore `cpio` archives from within the A/UX Startup utility. Because you do not have to be running A/UX to use this command, you can restore critical system files in the event of a serious system crash.
- You can restore system files directly from the *A/UX 3.0 CD-ROM*.
- It can copy special files, for example, everything stored in `/dev`. With `cpio`, a full backup includes all the files on the system (unlike the `tar` utility).
- By default, files are not restored on top of newer versions.
- When copying to floppy disks, `cpio` ejects disks that are full and prompts you to insert another disk.

The disadvantages of using `cpio` include:

- `cpio` archives may not be compatible among different computer systems unless you use the `-c` option for portability.
- The `tcbl` filter must be used when copying to cartridge tape drives, therefore you cannot tell `cpio` how much space is available on these tapes. This makes multiple-volume archives impossible.

The `cpio` utility copies files to or from the device or location you specify. For example, you can copy files to a tape or to a file system with equal ease. By default it copies from standard input and writes to standard output.

You do not give filenames directly as arguments to `cpio`. Instead, `cpio` takes its input from other utilities. The utilities most commonly used to give input to `cpio` are `find` and `ls`.

When using `ls` to provide input to `cpio`, do not use options because `cpio` must receive filenames at one per line. To copy only a few files, you can redirect the output of `ls` into a file, edit the file to remove unwanted filenames, and `cat` the file into `cpio` through a pipe.

The `cpio` utility typically restores files relative to the current directory. If you use relative pathnames when creating a backup with `cpio`, you have more flexibility about where you can restore files. However, if you use an absolute pathname, `cpio` will restore the file according to that pathname.

These are most common options used with `cpio`:

- o Copies *out* files to a device or location you specify, such as disk, tape blocking filter, or file.
- p Copies (*passes*) to another directory or file system you specify.
- i Copies *in* files from a device or location you specify, such as disk, tape, or file.
- t Lists the contents of the backup.

Refer to `cpio(1)` in *A/UX Command Reference* for a description of all of its options.

You will notice in later examples that, when `cpio` is used to copy to a floppy disk, the disk drive is referred to as a raw device rather than a block device. For example, you'll see

```
cpio -ov > /dev/rfloppy0
```

rather than

```
cpio -ov > /dev/floppy0
```

The `cpio` utility has been modified to take advantage of Macintosh disk drives and as a result gives faster performance on a raw device than on a block device.

`cpio` and tape backups

Most tape devices require that you send the data in 8K blocks. When you use `cpio` to copy to tape, you must use the `tcb` filter to block the data. The sole purpose of the filter is to block data in 8K blocks. Table 5-4 lists those devices requiring the use of the filter.

The `tcb` filter cannot recognize the end of the tape, nor can you specify to `cpio` how long a tape is. For this reason, a backup fails when `cpio` attempts to copy more files than will fit on one tape. If the command fails for this reason, you will see the following error message:

```
tcb: No such device or address
```

To complete the backup successfully, you need only to reenter the command and specify fewer files.

Table 5-4 `cpio` filter requirement

Backup device	Use filter?
Floppy disk	no
Apple 40SC	yes
8 mm DAT	yes
1/4" QIC	yes
4 mm DAT	yes
DCAS	yes
9-track	no

Here's an example of using the filter in a backup command:

```
ls | cpio -ov | tcb > /dev/rmt/tcX
```

The components of this command are as follows:

<code>ls </code>	The <code>ls</code> command lists the files in the current directory and sends them through a pipe (<code> </code>) as input to the <code>cpio</code> command.
<code>cpio -ov</code>	The <code>cpio</code> command, with options that copy out files (<code>o</code>), and display the filenames on the screen (<code>v</code>).
<code> tcb</code>	Pipes the files through the <code>tcb</code> filter, which blocks the data in 8K blocks, so that the tape unit can read it.
<code>> /dev/rmt/tc<i>x</i></code>	The output device, in this case the tape controller (<code>tc</code>) for a tape drive at SCSI ID <i>x</i> .

`cpio` and the A/UX 3.0 CD-ROM

The *A/UX 3.0 CD-ROM* stores the files in the root file system of the CD. Therefore, to access the files stored there with the `cpio` command, mount the CD on a directory and use the `-p` (pass) option of `cpio` to copy parts of the reference file system. See the section “CD-ROM and A/UX” in Chapter 6 for more details.

Using `cpio`

The examples shown in the following sections demonstrate various means of using the `cpio(1)` command. All the options mentioned in this section are available through the Commando interface as well. For a complete description of the command syntax, please see *A/UX Command Reference*.

Backing up specific files

To create selective backups—that is, to back up only those files that fall into a specified category—use the `find` command to pipe a list of files to `cpio`.

For example, to create a tape backup (to the tape device at SCSI ID 1) of all the files owned by the user `hobbes` anywhere in the file hierarchy, enter

```
cd /  
find . -user hobbes -print | cpio -ov | tcb > /dev/tc1
```

When using the `cpio` command to create backups, use the option to copy out (`o`) files. You may also want to display the name of every file copied on the screen using the verbose (`v`) option. If you want to ensure compatibility with other systems (and the `pax` utility) use the compatibility (`c`) option.

To create a floppy disk backup of all the files starting with the word `report` anywhere in the `/users` directory, enter

```
cd /
find users -name report\* -print | cpio -ovB > /dev/rfloppy0
```

When using the `cpio` command to create backups to floppy disks, use the option to block output (`B`) to 5120 bytes per record.

◆ **Note** You can use file expansion characters such as `*` and `?`, but these characters must be quoted (enclosed in single or double quotation marks or preceded by a backslash) to prevent the shell from interpreting them before they are passed to `find`. ◆

You can use other `find` options to select files by other characteristics, such as group ownership or age of the file. See also “Incremental Backups” later in this chapter and `find(1)` in *A/UX Command Reference*.

Backing up all files in a directory

You may often need to copy all the files in a directory. To make a backup of the `games` directory to floppy disk, use the following command:

```
cd /usr
find games -print | cpio -ovB > /dev/rfloppy0
```

To create a tape backup (to the cartridge tape device at SCSI ID 1) of the `games` directory, enter

```
cd /usr
find games -print | cpio -ov | tcb > /dev/tc1
```

You can use the `du(1)` command to estimate how much space a directory will take to back up. This command reports sizes in 512-byte blocks.

Creating partial backups

To create a partial backup, that is, to back up only files that have been modified or created since a certain time, use the `-mtime` option of the `find` command to select the files. For example, to backup (to the tape device at SCSI ID 1) all the files changed in the last 3 days, enter

```
cd /
find . -mtime -3 -print | cpio -ov | tcb > /dev/tc1
```

To make a floppy disk backup of user files that have changed in the last week, use the following command:

```
cd /users
find . -mtime -7 -print | cpio -ovB > /dev/rfloppy0
```

Creating multiple-archive backups

To create multiple archives on a single backup tape, use the no-rewind tape device to position the tape. For example, to create two archives on a tape (at SCSI ID 1) of all the files owned by the users `calvin` and `hobbes`, enter

```
cd /
find . -user calvin -print | cpio -ov | tcb > /dev/rmt/tc1n
find . -user hobbes -print | cpio -ov | tcb > /dev/rmt/tc1
```

By using the no-rewind device `/dev/rmt/tc1n` in the first command, the backups are placed sequentially on the tape. Because the second command uses the rewind device `/dev/rmt/tc1`, the tape rewinds after the archive is written. Unfortunately, in order to use multiple-archive backups effectively, you must keep a separate record of the archives written to tape and the order in which they were placed on the tape. No handy utility is available to automatically update a tape header with the backup information for multiple-archive tapes.

See the section “Extracting Files From Multiple-Archive Backups,” later in this section, for examples of how to skip between multiple archives on a single tape.

Listing the contents of the backup medium

To list the contents of a floppy disk archive made with `cpio`, use the command

```
cpio -it < /dev/rfloppy0
```

When using the `cpio` command to extract information from backups, use the option to copy in (`i`) files. To display the name of the files on the backup, use the table of contents (`t`) option.

To list the contents of a tape archive (assuming the tape drive is at SCSI ID 1) made with `cpio` and the `tcbl` filter, use the command

```
tcbl < /dev/rmt/tc1 | cpio -it
```

When extracting information from backups that have been written using the `tcbl` filter, you must first read in the files and unblock them (again using `tcbl`) then pipe them to `cpio`.

To list the contents of a 9-track tape archive (again, the tape drive is at SCSI ID 1) made with `cpio`, use the command

```
cpio -it < /dev/rmt/tc2
```

Extracting specific files from disk or tape

The process of extracting a file from a `cpio` archive is almost the reverse of the process used to create the archive. For example, to recover only certain files from a floppy disk or 9-track tape created with `cpio`, the command takes the form

```
cpio -iv filename... < /dev/backupdevice
```

where *filename...* is the name of the file or files to be extracted, and the device file `/dev/backupdevice` indicates on which device the archive can be found. This command line uses the verbose (`v`) option, but additional options can be used. The directory (`d`) option creates any directories needed to extract the files. The modification (`m`) option preserves the file's modification date. The update (`u`) option extracts files from the archive unconditionally. (Normally, `cpio` will not extract a file that is older than an existing file with the same pathname.) See `cpio(1)` in *A/UX Command Reference* for additional information.

◆ **Note** You can use the `-c` option of `cpio` to read archives backed up with the `pax` utility. ◆

For example, to extract the files starting with the word `report` from a floppy disk backup, use the command:

```
cpio -iv 'report*' < /dev/rfloppy0
```

◆ **Note** You can use file expansion characters such as `*` and `?`, but these characters must be quoted (enclosed in single or double quotation marks or preceded by a backslash) to prevent the shell from interpreting them before they are passed to `cpio`. ◆

To extract the files starting with the word `report` from a 9-track backup, use the command:

```
cpio -iv 'report*' < /dev/rmt/tc2
```

To extract the files starting with the word `report` from a cartridge tape backup, use the command:

```
tcb < /dev/rmt/tc1 | cpio -iv 'report*'
```

Extracting all files from disk or tape

To recover all files from a disk or tape created with `cpio`, you simply don't specify which files you want. In the absence of any filename specification when reading in files, `cpio` extracts the entire archive.

For example, to extract all the files from a floppy disk backup, use the command:

```
cpio -iv < /dev/rfloppy0
```

To extract the files from a 9-track backup, use the command:

```
cpio -iv < /dev/rmt/tc2
```

To extract the files from a cartridge tape backup, use the command:

```
tcb < /dev/rmt/tc1 | cpio -iv
```


Extracting files from multiple-archive backups

To extract files from multiple archives on a single backup tape, use the no-rewind tape device and the `mt` command to position the tape. For example, to recover a file named `testfindings` from the third archive on a tape, enter

```
mt -f /dev/rmt/tc1n rewind
mt -f /dev/rmt/tc1n fsf 2
tcb < /dev/rmt/tc1 | cpio -iv testfindings
```

The `mt` command is first used to rewind the tape (to make sure it is at the start of the tape) and then to position the tape, in this case it has been told to forward space the tape two end-of-file marks (`fsf 2`), positioning it at the beginning of the third archive on the tape. After the `cpio` archive is extracted, the tape rewinds because the rewind device was used in the last command.

In the event of hard I/O errors

When you're checking data after you have created backups, you must restart the entire procedure if a hard I/O error occurs while the data is being read. However, before beginning the procedure anew, try reinserting the problem disk or tape; often the system can read it the second time. If it can't, then you need to start the procedure from the beginning, using a fresh disk or tape.

You cannot interrupt `cpio` to allow formatting of additional tapes. If a cartridge requires manual formatting, the process must be stopped, fresh media formatted, and the procedure started again from the beginning. Therefore, have plenty of formatted media ready beforehand.

About `tar`

The utility `tar`, which stands for “tape archiver,” is used to back up directories or individual files in a file system. For complete information on this utility, refer to `tar(1)` in *A/UX Command Reference*.

The advantages of using `tar` include:

- `tar` accepts a blocking factor (with the `b` option) that lets you match the block size of the `tar` output with the block size of the output device. Typically this feature is used with tape drives.
- You can specify the maximum number of blocks a backup medium can hold.
- You can restore `tar` archives from within the A/UX Startup utility. Because you do not have to be running A/UX to use this command, you can restore critical system files in the event of a serious system crash.
- It is portable to systems that only have the `tar` utility.

The disadvantages of using `tar` are as follows:

- If you do not specify capacity and the copy runs off the end of the tape, you have to start the backup over again.
- It can copy only regular files and directories, not special files. Thus, a “full backup” made with `tar` may have files missing.

The `tar` command copies a single file or groups of files. As with `cpio`, `pax`, or `dump.bsd`, the files are copied sequentially; no directory structure on the backup medium is maintained.

When retrieving files, `tar` puts them into the current directory and keeps the directory structure you indicated. For example, if you change to the directory `/user/harvey` and copy the file `/user/harvey/stories/nickname` by giving the relative pathname `stories/nickname`, when you retrieve this file it will be copied into the current directory as `stories/nickname`.

If you use relative pathnames when creating a backup with `tar`, you have more flexibility about where you can restore files. However, if you use an absolute pathname, `tar` will restore the file according to that pathname.

If you might be restoring the files to an account other than the one from which you copied them (for example, to someone else’s account on another computer), then be sure to use relative pathnames.

The `tar` command has the capability of adding to files already on a floppy disk. It can also display a listing of the files archived on a particular disk or tape.

By default, `tar` writes to and reads from the file `/dev/mt/0m`. If you frequently use some other backup device, you can link this file to the device and save time.

`tar` and tape backups

By default, `tar` copies data in 512-byte blocks. This works fine when copying to floppy disks because the disks store information in blocks of that size. However, when copying to most tape drives, data must be sent in 8K blocks, and this size must be specified. Here is an example:

```
tar cbf 16 /dev/rmt/tc1 usr/lib
```

The components of this command are as follows:

<code>tar cbf</code>	This example shows the <code>tar</code> command with the option <code>cbf</code> . The <code>c</code> option creates a new backup, writing at the beginning of the tape. The <code>b</code> option alerts <code>tar</code> to use the following argument as the block size for sending out the files. The <code>f</code> option alerts <code>tar</code> to use the next following argument as the place in which to copy the files.
<code>16 /dev/rmt/tc1</code>	The block size used for most tape devices is 16, this is the argument for the <code>b</code> option. The device <code>/dev/rmt/tc1</code> is the argument for the <code>f</code> option.
<code>usr/lib</code>	This specifies the files to copy, in this example the directory <code>usr/lib</code> .

If a backup requires multiple volumes

When copying to a tape cartridge or to a floppy disk, `tar` does not recognize the end of the medium. It can run past the end of the tape or disk and generate an error message, forcing you to begin copying again. To protect against this annoying situation, let `tar` know the holding capacity of your backup medium if you suspect that your backup might require more than one tape or disk. Table 5-5 shows the holding capacity of various backup media, and the blocking and capacity arguments to use.

Table 5-5 `tar` blocking and capacity arguments (in blocks)

Backup media	Size	Blocking argument	Capacity argument (typical)
Floppy disk	800 KB		B 1600
	1.4 MB		B 2600
Apple 40SC		b 16	B 72000
8 mm DAT	15m tape	b 16	B 278320-556640
	54m tape	b 16	B 1113280-2319335
	112m tape	b 16	B 2319335-4638670
1/4" QIC		b 16	B 307198
4 mm DAT		b 16	B 2412541
DCAS		b 16	B 1113280
9-track	2400 ft. tape	d 6250	s 2300

Here's an example of how to let `tar` know the holding capacity of an Apple 40SC cartridge tape:

```
tar cbBf 16 72000 /dev/rmt/tc1 usr/lib
```

The `b` option alerts `tar` to use the argument `16` as the blocking factor. The `B` option alerts `tar` that 72,000 is the maximum number of 512-byte blocks the tape can hold. Note that the 8 mm DAT drives use a physical block size of 1024 bytes.

The exact number of blocks a tape can hold varies, depending on the number of potentially bad blocks on the tape. In this example, it is safe to use 72,000 blocks as the maximum tape capacity because this figure allows for the maximum number of bad blocks the space required to store the tape's formatting information (roughly 10% of the tape capacity). When a tape is formatted, potentially bad blocks are eliminated from the tape's usable space.

Copying to a disk

When copying data to a floppy disk using the `tar` command, you must enter the size of the backup medium in 512-byte blocks (see Table 5-5 for capacity arguments).

Additionally, you must use the eject floppy device (`/dev/fd/dxew`, where `x` is the drive number) to assure this utility ejects the disk when it is full. If you don't care about the floppy ejecting, you can use the regular floppy device (for example, `/dev/xfloppy0` or `/dev/fd/d0`).

◆ **Note** Because the `tar` command writes additional data on the disk to keep track of directory files, slightly less than the specified amount of data that a disk holds will be usable for storage. ◆

Because disks are random-access devices, there are several `tar` options available that cannot be used with tape drives. You can, for instance, add a file onto the end of the archive. Also, you can have multiple versions of the same file on a single disk.

Using `tar`

The examples shown in the following sections demonstrate various means of using the `tar(1)` command. All the options mentioned in this section are available through the Commando interface as well. For a complete description of the command syntax, please see *A/UX Command Reference*.

Backing up specific files

You will often need to save specific files that hold important data. Examples of such files include `/etc/passwd` and `/etc/group`. Because these files are crucial to the operation of the system, and because they may be frequently modified, they are vulnerable to corruption or even loss.

For example, to copy `/etc/passwd` and `/etc/group` to a disk, use the commands

```
cd /etc
tar cf /dev/fd/d0ew passwd group
```

When using the `tar` command to create backups, use the `c` option to create an archive; this starts a backup at the beginning of the medium. You can specify the backup device with the device (`f`) option.

To copy these files to an Apple Tape Backup 40SC tape drive, use the commands

```
cd /etc
tar cBf 16 72000 /dev/rmt/tc1 passwd group
```

To copy these files to a 9-track tape drive at 1600 bpi density, use the commands

```
cd /etc
tar cdsf 1600 2300 /dev/rmt/tc2 passwd group
```

Backing up an entire directory

When copying a directory, `tar` copies all its contents, including the contents of its subdirectories.

For example, to copy all the files in the home directory of the user `hobbes`, enter the commands

```
cd /users
tar cf /dev/fd/d0ew hobbes
```

To copy the user library files to an 8 mm DAT tape drive (onto a 15 meter tape), use the commands

```
cd /
tar cBf 16 278320 /dev/rmt/tc1 usr/lib
```

To copy these files to a 9-track tape drive at 6250 bpi density, use the commands

```
cd /
tar cdsf 6250 2300 /dev/rmt/tc2 usr/lib
```

◆ **Note** If you plan to place these files into a location other than the one you copied them from, be sure to use relative pathnames. Otherwise, if you use the absolute pathname, the files will retain that pathname and can be restored to only that pathname. You lose flexibility when you use absolute pathnames. ◆

Creating multiple-archive backups

To create multiple archives on a single backup tape, use the no-rewind tape device to position the tape. For example, to create two archives on a tape (at SCSI ID 1) of the home directories of the users `tony` and `cleo`, enter

```
cd /users
tar cbf 16 /dev/rmt/tc1n tony
tar cbf 16 /dev/rmt/tc1 cleo
```

In this example, no capacity argument was given, thus there is a risk the data will run off the end of tape. By using the no-rewind device `/dev/rmt/tc1n` in the first `tar` command, the backups are placed sequentially on the tape. Because the second `tar` command uses the rewind device `/dev/rmt/tc1`, the tape rewinds after the archive is written. Unfortunately, in order to use multiple-archive backups effectively, you must keep a separate record of the archives written to tape and the order in which they were placed on the tape. No handy utility is available to automatically update a tape header with the backup information for multiple-archive tapes.

See the subsection “Extracting Files From Multiple-Archive Backups,” later in this section, for examples of how to skip between multiple archives on a single tape.

Appending a file to a disk

You can append a file or files to an archive already on a floppy disk (but not to a tape archive). To copy the file `mvusr` from the current directory and append it to the files on a disk, enter

```
tar rf /dev/fd/d0ew mvusr
```

The replace (`r`) option appends the file to the `tar` archive on the disk.

Adding a later version of a file to a disk

To add a later version of the file `curses.mail` from the current directory to a floppy disk (but not to a tape), enter

```
tar uf /dev/fd/d0ew curses.mail
```

This command uses the update (`u`) option to add the named files to the disk if they are not already on the disk or if they are modified since they were last written to the medium.

The `tar` command responds with the message

```
a curses.mail 3 blocks
```

The `a` indicates that the file has been added to the archive. No message is printed, and no copy is made, if the file is identical to the copy in the archive.

Listing the contents of the backup medium

When using the `tar` command to list the files in backups, use the table of contents option (`t`).

For example, to list the contents of a `tar` archive on a floppy disk, enter the command

```
tar tvf /dev/rfloppy0
```

The `tar` command then displays the files with their permissions, ownerships (UID/GID), size in bytes, and dates:

```
rwxr-xr-x 102/202 978 Feb 1 14:16 1992 ./envelope
rwxr-xr-x 102/202 211 Apr 16 11:29 1992 ./proofread
rwxr-xr-x 102/202 978 May 10 10:34 1992 ./envelope
```

The same filename can appear more than once; `tar` allows multiple versions of a file on the same disk, but only if the versions are different.

To list the contents of a `tar` archive on a cartridge tape drive, use the command

```
tar tvBf 16 /dev/rmt/tc1
```

To list the contents of a `tar` archive on a 9-track tape drive, no blocking factor is necessary. Use the command

```
tar tvf /dev/rmt/tc2
```

Extracting a specific file or directory

To recover a specific file from a backup medium created with `tar`, use the following commands.

```
cd directory
```

```
tar xf /dev/backupdevice filename..
```

where *directory* specifies the directory into which you will restore the information, *backupdevice* specifies the device from which to read, and *filename*... specifies the file or directory to be extracted.

The `tar` utility extracts files into the current directory. Therefore, when you use `tar` to extract a file, first change directories to the target directory and specify the name of the file you want to extract as it appears in the `tar` listing of contents.

For example, to extract `/etc/passwd` and `/etc/group` files from a disk, use the commands

```
cd /etc
tar xf /dev/fd/d0e passwd group
```

To extract all the files from a `tar` archive on a cartridge tape drive, use the commands

```
cd /tmp
tar xBf 16 /dev/rmt/tc1
```

To extract all the files from a `tar` archive on a 9-track tape drive, use the commands

```
cd /tmp
tar xf /dev/rmt/tc2
```

In the absence of any filename specification when extracting files, `tar` extracts the entire archive.

The `tar` utility recovers the last stored version of a file from disk by overwriting all previous versions. For example, if a floppy disk archive holds several copies of the file `curses.mail`, to get the most recent copy you would enter

```
tar xf /dev/rfloppy0 curses.mail
```

The `tar` command responds with a message similar to

```
x curses.mail, 1135 bytes, 3 tape blocks
x curses.mail, 2036 bytes, 4 tape blocks
x curses.mail, 2042 bytes, 4 tape blocks
```

In this example, the `x` at the beginning of the line indicates that the file has been extracted. This example shows that each instance of the file is extracted; earlier versions of the file are successively overwritten and you are left with the last version on the disk.

Extracting files from multiple-archive backups

To extract files from multiple archives on a single backup tape, use the no-rewind tape device and the `mt` command to position the tape. For example, to recover a file named `projections` from the third archive on a tape, enter

```
mt -f /dev/rmt/tc1n rewind
mt -f /dev/rmt/tc1n fsf 2
tar xBf 16 /dev/rmt/tc1 projections
```

The `mt` command is first used to rewind the tape (to make sure it is at the start of the tape) then to position the tape, in this case it has been told to forward space the tape two end-of-file marks (`fsf 2`), positioning it at the beginning of the third archive on the tape. After the `tar` archive is extracted, the tape rewinds because the rewind device was used in the last command.

Extracting a particular version of a file

To recover a particular version of a file from disk, determine which version is needed by using the `t` and `v` options to display the contents of the disk or tape. See “Listing the Contents of the Backup Medium,” earlier in this section.

Once you determine which file you want to extract, use the `w` option with `tar`. When using the `w` option, you must confirm the action you tell the system to take. For example, to extract the May 10 version of `mvusr`, enter

```
tar xvwf /fd/d0ew ./mvusr
```

When `tar` displays the desired version of the file, enter `y`, as shown here:

```
x rwxr-xr-x 0/1 140 Mar  4 18:14 1992  mvusr:n
x rwxr-xr-x 0/1 162 Mar 10 10:46 1992  mvusr:y
x ./mvusr, 162 bytes, 1 tape blocks
x rwxr-xr-x 0/1   3 Mar 10 10:48 1992  mvusr:n
```

A `y` (yes) response indicates that the file should be extracted. A response of `n` (no) or `RETURN` means the file should not be extracted. If you are unsure which version you want, you can use the `pax` utility with the `-i` option to read the `tar` archive and interactively rename the files as they are extracted.

About `dump.bsd` and `restore`

The `dump.bsd` command is used for partial or full file system backups. The utility supports *dump levels*; these levels range from 0 to 9, where 0 represents a full backup of an entire file system and 1 through 9 are used for partial file system backups.

The `restore` program is the companion utility of `dump.bsd`. It retrieves files and directories from a backup medium created with `dump.bsd`.

The advantages of using `dump.bsd` and `restore` include the following:

- `dump.bsd` is generally faster than `tar` or `cpio`.
- They allow you to back up only those files modified or created after a certain date. The `dump.bsd` utility does this by keeping a record of the last `dump` date for each level, and using that record to determine those files created or modified since that date.
- The `restore` command can extract files by their inode numbers. This feature can be helpful if the file checking program, `fsck`, has removed corrupted files and only informed you of the inode number of the file it removes.
- A variation of the `dump.bsd` command, `rdump`, allows you to back up files over a network. See *A/UX Network System Administration* for details.
- You cannot inadvertently back up a file system located on a remote computer.

The disadvantages of using `dump.bsd` and `restore` are as follows:

- They operate only on file systems. As distributed, A/UX has only one file system, so this is not initially a drawback. However, if you add one or more file systems, your use of these backup utilities becomes more complicated because you have to back up the file systems individually.
- You must be sure the system's date and time are always correct, or you are likely to lose files when restoring from an incremental backup.
- Backups made with `dump.bsd` cannot usually be transferred to other systems.

For complete information on these commands, refer to `dump.bsd(1M)` and `restore(1M)` in *A/UX System Administrator's Reference*.

The `dump.bsd` and `restore` backup utilities are recommended for multi-user installations. When you use `dump.bsd` to create a backup, you must use `restore` to restore a file, directory, or file system.

◆ **Note** The `/etc/dumpdates` file must exist; otherwise `dump.bsd` displays the error message `/etc/dumpdates: No such file or directory`. Therefore, before you run `dump.bsd` for the first time on your system, enter the command `touch /etc/dumpdates` ◆

By default, `dump.bsd` writes to, and `restore` reads from, the file `/dev/tape`. If you frequently use some other backup device, you can link `/dev/tape` to the device you use and save time.

Dump levels

When you use the `dump.bsd` command, you specify an incremental backup using dump levels, which are integers that can range from 0 through 9. Instead of specifying a date to indicate that you want to back up everything that has been created or modified since that date, you specify a dump level to indicate that you want to back up everything that has been created or modified since you made a backup with a lower dump level.

For example, a level 7 dump backs up all files modified since the most recent backup at dump level 6 or lower. Thus dump level 0 represents a full backup.

Using dump levels in a monthly backup strategy

Most multi-user installations use a dump strategy based on once-a-month full dumps:

- Every month do a full dump (level 0):
`dump.bsd 0u filesystem...`
- At the end of each week do a weekly dump (level 4):
`dump.bsd 4u filesystem...`
- At the end of each working day do a daily dump (level 7):
`dump.bsd 7u filesystem...`

If no dump level is specified, `9u` is assumed. Many keys can alter the default operation of the `dump.bsd` command. See the man page `dump.bsd(1M)` in *A/UX System Administrator's Reference*.

The level numbers can be used mnemonically to represent weeks (4 weeks in a month) and days (7 days in a week). In this strategy, the weekly dump backs up all files modified since the last monthly backup, and the daily dump backs up files modified since the last weekly dump. With this scheme for routine backups you need to use only three dump levels to restore an entire file system to its most recent backup.

For the daily and weekly backups, it is suggested that you use at least two sets of backup media. For monthly (level 0) dumps, use a set of fresh disks or tapes and save them for an extended period (generally 6 months to a year). See the section “A Common Backup Scheme,” earlier in this chapter, for a backup strategy. A different, more secure strategy, is given in `dump.bsd(1M)`.

`dump.bsd`, `restore`, and tape backups

The combination of programs `dump.bsd` and `restore` were written to facilitate multivolume dumps of large amounts of data. By default, `dump.bsd` copies data out in 512-byte blocks. While this works for floppy disks, data being sent to tape drives must be sent in larger blocks and this block size must be specified. The block size varies with the type of drive used for backups, as shown in Table 5-6. This table also shows ranges for some capacities because different models of tape drives can record different amounts of information on the same length tape.

Table 5-6 `dump.bsd` blocking and capacity arguments

Backup media	Size	Blocking argument	Capacity argument
Floppy disk	800 KB	F	
Apple 40SC		c	
8 mm DAT	15m tape	b 16b	s 300m-600m
	54m tape	b 16b	s 1200m-2500m
	112m tape	b 16b	s 2500m-5000m
1/4" QIC		b 16b	s 150m-525m
4 mm DAT		b 16b	s 2000m
DCAS		b 16b	s 600m
9-track	2400 ft. tape	■ 6250	s 2300f

Capacity arguments to `dump.bsd` can be specified in blocks, kilobytes, megabytes, or feet. For additional information, see `dump.bsd(1M)`. The `restore` program does not need blocking or capacity arguments; it is capable of detecting both and adjusting itself accordingly.

Using `dump.bsd`

The `dump.bsd` command operates on the file system—UFS or SVFS—residing on a specified disk partition. It copies all files modified after a certain date to a floppy disk or other backup medium.

When disks are used to create backups, `dump.bsd` sets a checkpoint for itself at the beginning of each disk. If some error occurs during writing to a disk, `dump.bsd` waits until you have removed the old disk and inserted a new one, then (after prompting with a question) restarts itself from the checkpoint.

The `dump.bsd` utility prompts with questions when

- it reaches the end of a disk
- it reaches the defined capacity
- it reaches the end of a dump
- a hard I/O error occurs

You must enter either `yes` or `no` to all of the questions asked by `dump.bsd`.

Full backups

Full system backups are generally made to tape rather than floppy disks because of the huge amount of data involved. The following example shows the `dump.bsd` command used to dump the contents of an entire file system located on SCSI disk 4, slice 0, to a cartridge tape drive (at SCSI ID 1) having a capacity of 300 megabytes using a blocking factor of 8 kilobytes:

```
dump.bsd 0ubsf 8k 300m /dev/rmt/tc1 /dev/rdisk/c4d0s3
```

- ▲ **Caution** Be careful not to transpose the name of the device being written (the first filename argument) and the name of the device being read (the second filename argument). An empty file system is the likely result of such an action. ▲

When using the `dump.bsd` command to backup an entire file system, use the following options. The option `0` specifies a complete backup (not incremental). The update (`u`) option updates the system file `/etc/dumpdates` with the time of this backup. The blocking (`b`) option specifies the blocking factor, in this example 8 kilobytes. The space (`s`) option defines the amount of space on the backup media, here 300 megabytes is specified. The file (`f`) option tells `dump.bsd` to write the dump on a device other than the default device, in this case the tape drive at SCSI address 1. The last argument, `/dev/rdisk/c4d0s3`, specifies the file system to be backed up.

To dump this same file system to a 6250 bpi 9-track tape drive (at SCSI ID 2) having a 2400 foot tape on it:

```
dump.bsd 0udsf 6250 2300f /dev/rmt/tc2 /dev/rdisk/c4d0s3
```

- ◆ **Note** If the name of the device you are backing up to is `-`, `dump.bsd` writes to the standard output, in which case `dump.bsd` can be used as part of a pipeline. ◆

Incremental backups

Incremental backups can often be made to floppy disks because the amount of data is much smaller than for a full system backup. The following example shows the `dump.bsd` command used for a daily backup:

```
dump.bsd 7uFf /dev/rfloppy0 /dev/rdisk/c0d0s0
```

When using the `dump.bsd` command to make incremental backups, use the following options. The option `7` specifies an incremental. The floppy disk option (`F`) option sets the blocking and space parameters to those for a double-density floppy disk.

Finding out which file systems to back up

You can have `dump.bsd` check which file systems need to be backed up, as defined in the file `/etc/fstab`. Each file system listed in this file has a “dump frequency” field that specifies, in days, how often to back up the file system. See Figure 8-6 for an illustration of this file’s contents. Issue the command

```
dump.bsd w
```

A related option, `w`, (note that this is a capital *w*) gives additional information such as the date and level of the last backup.

Using `restore`

The `restore` command reads the backup media created with the `dump.bsd` command and restores files into the current directory. When using `restore`, you must be careful not to replace the current file system with an older version of itself. This command provides support for recovering files from multiple-archive tapes.

If the archive spans multiple volumes, `restore` expects to read starting from the first volume of the archive.

Listing the contents of the backup medium

Individual files or directories can be located on `dump.bsd` backups by listing the contents of the medium.

To list the files on a floppy disk, for example, you might use a command such as

```
restore -tFf /dev/rfloppy0
```

The table of contents (`t`) option tells `restore` to list the files on the backup medium.

To list the files on a backup tape, you might use a command like:

```
restore -tf /dev/rmt/tc1
```


The `restore` utility reads the tape to determine the blocking factor, then lists the files on the tape.

To list the files on the third archive of a multi-archive backup tape, use a command like:

```
restore -tfs 3 /dev/rmt/tc1
```

The `-s` option provides direct support for multiple archives, so you can avoid using the `mt` command to position the tape.

Restoring individual files

Individual files or directories can be restored from `dump.bsd` backups by specifying the filenames desired.

◆ **Note** The `restore` command extracts files and places them into the current working directory. Before restoring any files, make sure you are in the appropriate directory. ◆

To extract a file from an incremental backup on a floppy disk, for example, you might use a command such as

```
restore -xFf /dev/rfloppy0 calendar "trip report"
```

The extract (`x`) option tells `restore` to copy files from the backup medium. This example is restoring two files, named `calendar` and `trip report`. Note that the second file name must be enclosed in quotation marks (either single or double) to avoid confusing the shell.

To extract these same files from a backup tape, you might use a command such as

```
restore -xf /dev/rmt/tc1 calendar "trip report"
```

The `restore` utility reads the tape to determine the blocking factor, then extracts the named files.

Restoring a full backup

Generally there are two reasons for needing to restore a full backup: catastrophic failure, and moving a file system to a new disk. The following procedure works for either situation.

1 Start A/UX.

If you have suffered major failure, follow the procedure given in the section “Using Autorecovery” in Chapter 6 to get a minimum A/UX running.

2 Log in as root.

3 Bring the system to single-user state.

Enter the command:

```
shutdown
```

Wait for the A/UX Finder to be shut down and the single-user window to open.

4 Mount the file system you are restoring if you are restoring a file system other than the root file system.

Enter the command:

```
mount /dev/dsk/cxdj5z /mnt
```

5 Change directory to the mount point if you are restoring a file system other than the root file system.

Enter the command:

```
cd /mnt
```

6 Insert the first volume of your latest level 0 backup.

Wait for the drive to become ready.

7 **Restore your latest level 0 backup.**

Enter the command:

```
restore -rf /dev/backupdevice
```

The `restore` program will prompt you if you need to insert additional volumes.

- ▲ **Warning** The `-r` option should be used only to restore a complete `dump.bsd` archive onto an empty hierarchy or to restore an incremental `dump.bsd` archive after a full level 0 restore. Be very careful about where you are in the file system when you use the `-r` option. If you start `restore -r` from the top of a full hierarchy, the system replaces current files with older versions. ▲

8 **Insert the first volume of your latest next-lowest-level backup.**

Wait for the drive to become ready. If you followed the suggestion in the section “Using Dump Levels in a Monthly Backup Strategy,” you would use your latest level 4 backup.

9 **Restore your latest next-lowest-level backup.**

Enter the command:

```
restore -rf /dev/backupdevice
```

The `restore` program prompts you if you need to insert additional volumes.

10 **Repeat steps 8 and 9 until you have restored your latest daily backup.**

If you followed the suggestion in the section “Using Dump Levels in a Monthly Backup Strategy,” you would repeat the steps once, using your latest level 7 backup.

11 **Remove the file** `/restoresymtab`.

This file is used to store information about the level of backups that have been recovered for each file system.

12 **Make a new level 0 backup.**

Once a full backup has been restored, a new level 0 backup must be made immediately. Restoring changes file inodes, thus the prior incremental backups are no longer reliable.

13 **Change directory to the root directory if you are restoring a file system other than the root file system.**

Enter the command:

```
cd /
```

14 **Unmount the file system you are restoring if you are restoring a file system other than the root file system.**

Enter the command:

```
umount /mnt
```

15 **Bring the system to multi-user state.**

Enter the command:

```
init 2
```

After this procedure, the file system is fully restored.

Interactive mode for `restore`

The `restore` command features an interactive mode for extracting files from a `dump.bsd` backup. This version of the command lets you navigate a tape as though it were a disk, adding filenames to a list of files to be restored. To invoke the interactive mode, use the `-i` option:

```
restore -if /dev/backupdevice
```

When you use `restore` in the interactive mode, it reads directory information from the backup medium and then creates a shell-like interface, complete with the following prompt:

```
restore >
```

This interface lets you move around the directory tree, adding files into a list to be extracted. The interface also supports commands that aid in locating files.

The available commands are:

```
add          ls
cd           pwd
delete      quit
extract     setmodes
help        verbose
```

If a command needs an argument and one is not provided, the current directory is used by default. These commands are fully described under `restore(1M)` in *A/UX System Administrator's Reference*.

A sample session is shown below. For this example, the administrator is restoring the home directory of the user `frank`, which was accidentally removed. The responses of the administrator are shown in **bold**.

```
wintermute.root# restore -if /dev/rmt/tcl
restore > pwd
/
restore > cd users/frank
restore > ls
memos
games
reports
restore > cd /users
restore > add frank
restore > extract
You have not read any tapes yet.
Unless you know which volume your file(s) are on you should
start with the last volume and work towards the first.
Specify next volume#: 1
set owner/mode for '.'?[yn] n
restore > quit
```

See `restore(1M)` in *A/UX System Administrator's Reference* for details.

About `pax`

The `pax` command is a utility defined by POSIX, an international standards organization working to find portable operating system standards derived from UNIX. (POSIX stands for IEEE Standard Portable Operating System Interface for Computer Environments.) Although the archives it creates differ slightly from those created by `tar` and `cpio`, these archives conform to POSIX standards. See `pax(1)` in *A/UX Command Reference* for a description.

The utility `pax` backs up and restores an entire directory hierarchy or individual files. The advantages of using `pax` include:

- It can copy device files and symbolic links.
- `pax` can detect the end of media on floppy disks and 9-track tape drives.
- It can read archives created by both `tar` and `cpio`.

The disadvantages of using `pax` include:

- `pax` archives are not always supported by different computer systems.
- The `tcB` filter must be used when writing to cartridge tapes.

The `pax` command copies files to or from the device or location you specify. For example, you can copy files to a tape or to a file system with equal ease. By default it copies from standard input and writes to standard output.

`pax` and tape backups

By default, `pax` copies data out in 512-byte blocks. This works fine when copying to floppy disks because the disks store information in blocks of that size. Because the 9-track drive have a variable block size, no blocking argument is needed. In order to write to cartridge tape drives, you must use the `tcB` blocking filter.

Using `pax`

The examples shown in the following sections demonstrate various means of using the `pax(1)` command. All the options mentioned in this section are available through the Commando interface as well. For a complete description of the command syntax, please see *A/UX Command Reference*.

Backing up specific files

To back up only those files that fall into a specified category, use the specification as the last argument on the `pax` command line. Wildcard characters can be used; they are expanded by the shell before being passed to the `pax` command.

For example, to create a backup to floppy disk of all files in the current directory whose names begin with `report` or `rpt`, use

```
pax -w -t /dev/rfd/d0e report* rpt*
```

When using the `pax` command to create backups, use the `w` option to write files. You usually must specify the backup device to use with the device (`t`) option. The shell expands the wildcard characters before passing the arguments to the command.

To create a 6250 bpi backup to a 9-track tape drive (at SCSI address 2) of all files in the current directory whose names begin with `report` or `rpt`, use

```
pax -w -t /dev/rmt/tc2.6250 report* rpt*
```

To create a backup to a cartridge tape drive (at SCSI address 1) of all files in the current directory whose names begin with `report` or `rpt`, use

```
pax -w report* rpt* | tcb > /dev/rmt/tc1
```

Backing up all files in a directory

Backing up directories is similar to backing up files; you can either specify the directory name, or change to the directory and copy all the files with wildcards.

For example, to back up the directory `/users` using a 9-track tape drive (at SCSI address 2), you might use the following commands:

```
cd /  
pax -w -l -t /dev/rmt/tc2 users
```

In order to save tape space with large backups, you can use the option to link (`l`) files rather than copying them (wherever possible).

To back up the directory `/users` using a cartridge tape drive (at SCSI address 1), you might use the following commands:

```
cd /  
pax -w users | tcb > /dev/rmt/tc1
```

To create a backup to floppy disk of all files in the home directory of the user `calvin`, use

```
cd /users  
pax -w -t /dev/rfd/d0e calvin
```

Listing the contents of the backup medium

To list the contents of a floppy disk backup made with `pax`, enter

```
pax -t /dev/rfd/d0e
```

To list the contents of a 9-track tape backup (for a drive at SCSI address 2), use

```
pax -t /dev/rmt/tc2
```

To list the contents of a cartridge tape backup (for a drive at SCSI address 1), use

```
tcb < /dev/rmt/tc1 | pax
```

Extracting specific files from a disk or tape

To recover only certain files from a disk or tape created with `pax`, use `pax` with the read (`r`) option. You may want to first obtain a list of the full pathnames of files on the disk or tape.

For example, to extract the files whose names begin with `report` from floppy disk into the current directory, use

```
pax -r -t /dev/rfd/d0e report\*
```

To extract a backup from a 9-track tape drive (at SCSI address 2) of the files whose names begin with `report`, use

```
pax -r -t /dev/rmt/tc2 report\*
```


To extract a backup from a cartridge tape drive (at SCSI address 1) of the files whose names begin with `report`, use

```
tcb < /dev/rmt/tc1 | pax -r report\*
```

When extracting a file from a multivolume dump, `pax` reads through the entire volume set, even if the file you are extracting is on the first volume.

◆ **Note** When extracting files from multivolume backups, `pax` prompts you for the next volume, but doesn't check that the volume is the correct one. ◆

Extracting all files from a disk or tape

To recover all files from a disk created with `pax`, enter

```
pax -r -t /dev/fd/d0ew
```

In the absence of any filenames, `pax` reads the entire backup.

To recover all files from a 9-track backup created with `pax`, enter

```
pax -r -t /dev/rmt/tc2
```

To recover all files from a cartridge tape backup created with `pax`, enter

```
tcb < /dev/rmt/tc1 | pax -r
```

Verifying data on backed-up disks

To increase confidence that your backups contain accessible data, you can test them before you put them into storage. The `dd` command can be used to test all backup archives to find hard input/output (I/O) errors. If no hard I/O error is detected, the appropriate option for each backup utility can be used to list the contents. Reading with `dd` forces a read of the entire backup medium. Different backup media require different block size (`bs`) arguments for optimal performance; these are listed in Table 5-7.

Table 5-7  blocking arguments

Backup media	Size	Blocking argument
Floppy disk	800 KB	90b
	1.4 MB	90b
Apple 40SC		8k
8 mm DAT		8k
1/4" QIC		8k
4 mm DAT		8k
DCAS		8k
9-track	2400 ft. tape	none

For example, for floppy disks, insert each disk and enter

```
dd if=/dev/rfloppy0 of=/dev/null bs=90b
```

If the data is successfully read, messages like those that follow appear:

```
17+1 blocks in
```

```
17+1 blocks out
```

To read a 9-track tape (at SCSI ID 2), enter

```
dd if=/dev/rmt/tc2 of=/dev/null
```

To read a cartridge tape (at SCSI ID 1), enter

```
dd if=/dev/rmt/tc1 of=/dev/null bs=8k
```

If messages appear indicating hard I/O errors on any of the media, you need to restart the entire backup, using new media to replace the faulty ones. See `dd(1)` for additional information.

Automating system administration with `cron`

A variety of A/UX system administration tasks need to be done periodically. Some of these tasks, such as backups onto removable media, may require manual intervention. Most, however, can be run automatically by the A/UX `cron` daemon.

There are several benefits to using `cron`. For example, `cron` cannot forget to perform its functions, as a human might; it never goes on vacation or takes sick leave, and it is quite willing to do things at awkward times, such as 2:00 A.M. Also, `cron` can do things very frequently, as often as once a minute. Of course, in order for `cron` to complete its assignments, the system must be left running.

Instructions to `cron` can consist of single commands or shell scripts. The commands, and the definition of when the commands are to be performed, are located in the file `/usr/spool/cron/crontabs/user`, where *user* is the login name of the user on whose behalf the command will be run. These files are called *crontab files*. For example, the crontab file `/usr/spool/cron/crontabs/adm` controls timing of the various accounting functions. The format for defining the time a command is to be run is described in the section “The Crontab File Format,” later in this chapter.

User access to `cron`

Once an activity has been chosen for automation via `cron`, the system administrator must decide who is to perform the action. The `cron` daemon accords a command the same privileges as the user running the command. Since a user logged on as root has so few restrictions, `cron` commands should typically be put in the crontab file of some less powerful account.

To allow users to use the `cron` facility, add the names of users allowed to use `cron` to the file `/usr/lib/cron/cron.allow`. Users can be either user login names, such as `joe`, or administrative login names, such as `lp`.

If `cron.allow` is empty or absent because you’ve deleted it, then the system checks a file called `cron.deny`. This file lists users who are denied access to `cron`; again, you enter names into the file. It is your choice whether to use `cron.allow` or `cron.deny` to control access to `cron`. You would not use both. If neither file exists, only the root account is allowed to use `cron`.

In general, using the `cron.allow` file is more secure than using `cron.deny`, because the administrator must actively approve each user who might wish to use `cron`. If it is more convenient to use `cron.deny` and an administrator feels that a site's security needs are low, this option is available. Lack of both files is appropriate for sites where the root account runs `cron` commands on behalf of other users.

The crontab file format

The `cron` utility uses crontab files to control the execution of programs. Each line of text in this file consists of six fields that together specify a process to be executed at a certain time. The fields are separated by spaces or tabs. The first five fields specify a time. They specify, in order,

<i>minute</i>	Time in minutes (0-59)
<i>hour</i>	Time in hours (0-23)
<i>day-of-month</i>	Day of the month (1-31)
<i>month-of-year</i>	Month of the year (1-12)
<i>day-of-week</i>	Day of the week (0-6, with 0 as Sunday)

Each of these fields may contain either a list of elements separated by commas, which specify specific cases to be activated, or an asterisk (*), which means ignore all cases. An element may be one number or two numbers separated by a hyphen (-), meaning an inclusive range.

You specify days in two fields: *day-of-month* and *day-of-week*. If you specify both, both are used. To specify days using only one field, set the other to *.

The sixth field is the process to be executed. A percent sign (%) in this field (unless masked by \) is translated as the signal for a new line. The shell executes only the first line (up to the % or the end of the line).

For example, the line

```
15 3 1 * * find / -mtime -31 -print | cpio -ov | tcb > /dev/rmt/tc1
```

means this: At the fifteenth(15) minute of the third (3) hour on the first (1) day of each month, no matter what month of the year or day of the week the first day is, back up all the files that have changed in the last 31 days to the tape drive `/dev/rmt/tc1`. The asterisks in the *month-of-year* and *day-of-week* fields tell the system to ignore these fields.

Adding `cron` commands

Any allowed user can schedule commands to be run by `cron`.

1 Test the desired command, using a fresh login session.

You can use the substitute user command (`su user`) for the account that `cron` will emulate. Alternately, you can log in as *user*.

2 Execute the desired command just as it will appear in the crontab file.

This ensures that the permissions, command format, and user environment are the same as those used by `cron`. If the command fails, try running the command again when logged on as root. If the command works, the problem is that the user does not have the correct permissions to perform the task. If the command fails when you logged on as root, there is something wrong with the command itself.

3 Insert the command in a file with a specification of when `cron` is to run the command

See the earlier section “The Crontab File Format,” for information on how to define the time the command is to be run.

4 Install the file into the user’s crontab file.

While logged in as *user* (or impersonating *user* via `su`), enter the command:

```
crontab file
```

where *file* is the name of the file you created in step 4. This copies the file into `/usr/spool/cron/crontabs/user` and alerts `cron` to reread the crontab file. The *user* is determined by your login name or the login name you have changed to using the `su` command, whichever is most recent.

Monitor the results of a few executions of the automated activity to make sure the system is working smoothly.

Changing `crontab` commands

In order to change `crontab` commands, you replace the crontab file with a new version.

1 Copy the current crontab file.

Enter the command:

```
crontab -l > snapshot
```

to put a copy of the current `crontab` file into the file *snapshot*.

2 Edit the file *snapshot* to reflect the new changes.

3 Reinstall the file into the user's crontab file.

Enter the command:

```
crontab snapshot
```

where *snapshot* is the name of the file you created in step 1. This replaces the file `/usr/spool/cron/crontabs/user` and alerts `crontab` to reread the crontab file. The *user* is determined by your login name or the login name you have changed to using the `su` command, whichever is most recent.

Removing `crontab` commands

You can completely remove a crontab file using the command:

```
crontab -r
```


This removes the file `/usr/spool/cron/crontabs/user`. The *user* is determined by your login name or the login name you have changed to using the `su` command, whichever is most recent.

6 Managing Disks

About autorecovery / 6-3

Reclaiming disk space / 6-11

CD-ROM and A/UX / 6-16



Many sizes and makes of hard disks are available for use with A/UX. Regardless of make or size, any disk can become full, requiring you to provide more disk space. If A/UX does not have sufficient space, the system may cease to function correctly until space is made available. Also, any disk may fail, requiring you to recover data or lose it. This chapter suggests various means to free up space on disks to make room for user files. This chapter contains an example showing you how to use CD-ROM drives to help save hard disk space.

One way to prepare your system in the case of disaster is to expend a little effort to maintain the autorecovery system.

About autorecovery

The autorecovery feature of A/UX is designed to protect you from sudden, catastrophic loss of data and to minimize the need for a technical expert to diagnose and repair system problems. Autorecovery is one of the three methods available to repair your system if problems arise. The first is a standard file-consistency check (`fsck`), which is capable of repairing many file system problems. The second method is autorecovery, used to replace certain files that have become corrupted to the point the first method can't fix them. If autorecovery can't fix the error you can use the third method; using the Installer to rebuild your system and recover files from your backups.

Before the system is started, autorecovery identifies and compensates for bad disk blocks, file-system inconsistencies, and missing or damaged files. It performs this task by checking the file systems against parallel files in an autorecovery partition. If you engage in periodic maintenance, you can use autorecovery to rebuild your system after a system crash.

Autorecovery does have limitations: It is concerned only with critical system files, and it does not restore damaged or missing user files. You must keep backup copies of your own files in case you need to restore them.

Although most of its operation is automatic, you must perform some administration tasks to keep autorecovery up to date, just in case it is ever needed. This section describes those administration tasks.

- ▲ **Warning** Do not use autorecovery as a backup system, or add personal files to the autorecovery file system. ▲

Overview

Autorecovery refers both to the procedure that checks and repairs the A/UX file systems and to the special disk file systems used by this procedure. (For historical reasons, the term *eschatology* is still used in filenames and command names and in the arguments to commands although the term has been replaced by *autorecovery*. The terms *autorecovery* and *eschatology* refer to the same A/UX feature.)

One partition on the standard A/UX distribution disk is reserved for autorecovery. This area is a distinct A/UX file system, occupying 3 MB, that contains copies of key system files and other information about A/UX. If a key system file is damaged or destroyed, autorecovery copies the file from this file system to the A/UX root file system.

The autorecovery facility uses a list of key system files contained in the configuration master list (CML). The CML appears in the A/UX root file system in the file `/etc/cml/init2files`. A copy of this file also appears in the autorecovery file system.

Autorecovery can verify the physical condition of the disk and check each file in the CML when you start up the system. (By default, only a file system consistency check is done.) The autorecovery feature uses rules stored in the CML to check file attributes, such as size, ownership, permissions, type, modification time, version, and checksum. If any attributes do not match, autorecovery corrects the file attributes, if possible. If autorecovery cannot make these corrections, it replaces the file.

Because autorecovery depends on the CML, you must keep the CML up to date. When you add or change key system files, you must update the autorecovery files, using the autorecovery utilities `eu` and `eu•date`. The autorecovery facility is of little value unless a conscientious system administrator keeps the CML and autorecovery file system updated. The updating of the CML can be automated; for information on how to do this, see the section “Automating System Administration with `cron`” in Chapter 5.

Using autorecovery

Autorecovery is run from A/UX Startup under the Macintosh OS. The following procedure tells you how to invoke autorecovery:

- 1 Double-click the A/UX Startup icon.**

The A/UX Startup icon is found in the MacPartition disk icon.

- 2 Press COMMAND-PERIOD (or click the Cancel button) immediately.**

3 **Invoke the autorecovery command.**

Enter the command:

```
esch -bv
```

This command begins the autorecovery process. If your system is minimally damaged, this step will take a few minutes. If your system has sustained major damage, this step may take as long as 30 minutes. If you receive error messages, consult the index of this guide, under the heading “Error Messages,” to determine your next step.

4 **Start up A/UX.**

Enter the command:

```
boot
```

These steps should restore your system to functionality, provided that autorecovery was able to fix or restore the damaged files.

You can also configure the system to run autorecovery every time A/UX starts up; see Chapter 2, “System Startup and Shutdown,” for more information. For an explanation of the `esch` command and its options, see `esch(8)` in *A/UX System Administrator's Reference*.

How autorecovery works

The autorecovery feature of A/UX proceeds through several phases. First it examines the system information in its own file systems to verify that a system failure did not interrupt the previous invocation of autorecovery. If the autorecovery file system is suspect, autorecovery cannot use it to restore damaged or missing files in the A/UX root file system. If autorecovery detects that the autorecovery file system itself was being updated when the system failed, it will not attempt restoration.

The autorecovery feature then uses a version of `fsck` to check the A/UX root file system and the autorecovery file system for consistency. It attempts to correct any errors it finds. (See Chapter 8, “Checking the A/UX File System: `fsck`,” for a description of the file system checking routine.) If a file system is not repairable, autorecovery will, as a last resort, build a new file system in place of the damaged one and restore the files in the CML. When autorecovery remakes a file system, all data not in the CML is lost and must be restored from backups or reinstalled from the *A/UX 3.0 CD-ROM*.

Autorecovery administration

In order to keep the autorecovery file system up to date, you must perform two key autorecovery administration tasks whenever key system files change:

- Update the CML.
- Update the autorecovery copies of the files.

The system files that are most important to update when they change are:

```
/etc/fstab          /etc/passwd
/etc/group          /etc/profile
/etc/hosts         /unix
/etc/inittab
```

You perform the autorecovery administration tasks with the `eu` and `eupdate` utilities, described in this section and in *A/UX System Administrator's Reference*.

The `eu` utility updates the CML and the autorecovery file system. (A related utility, `escher`, also updates the CML but does not create checksums for the files. For this reason, use of `escher` is discouraged.) The `eupdate` utility copies the system files typically updated by autoconfiguration to the autorecovery file system and updates the CML entries for these files.

The `eu` utility

The `eu` utility updates the CML and copies a specified file to the autorecovery file system. The `eu` utility operates on one file at a time. To run `eu`, log in as root and enter the command

```
eu pathname
```

where *pathname* is the absolute pathname of the file to be copied.

The `eu` utility updates the CML even if the CML already contains an entry for the specified file. This utility can also be used to add entries to the CML, but the space in the autorecovery partition is so limited that this is not a good idea.

The `eupdate` utility

The `eupdate` utility updates the CML entries for the files typically modified when you set your system up for networking. To run `eupdate`, log in as root and enter the command

```
eupdate
```

The A/UX kernel (`/unix`) and other files necessary for multi-user network operation are copied to the autorecovery file system. The CML entries for these files are also updated.

Troubleshooting

This section discusses problems that occasionally occur when you run autorecovery. Although autorecovery is fairly robust, it errs on the side of caution. It will not replace damaged or missing files if the copy in the autorecovery file system, or the autorecovery file system itself, is suspect. This section presents the procedures to use when manual intervention is required to correct the operation of autorecovery.

◆ **Note** The procedures outlined here are for emergency use only. Normally, only autorecovery has access to the autorecovery file system. Manual intervention should be kept to a minimum. The procedures recommended here use the `pname` utility. Users unfamiliar with this utility should see the `pname(1M)` man page in *A/UX System Administrator's Reference* before continuing. ◆

A/UX Startup errors

The following errors can occur while A/UX is starting up.

```
warning. Inconsistent mount times in bzb.
```

This message means that the autorecovery partition has inconsistent mount times.

```
esch: no consistent type FSTEF SBZB (ES_BZBS_FSTEF S)  
[error occurred in Block Zero Block module]
```

If you get this message, autorecovery has failed during startup.

Occasionally, autorecovery may be unable to restore damaged or missing files, usually because the autorecovery file system was mounted when the system crashed. In this case, the autorecovery facility does not use this file system because the integrity of the data is not guaranteed.

The most obvious sign that autorecovery has been interrupted is a message that autorecovery has failed during startup. The autorecovery facility verifies that the unmount time is later than the mount time for the autorecovery file system. If the autorecovery file system has been left mounted, this test will fail, resulting in one of the startup warning messages. The autorecovery file system must be checked for integrity. Use the following procedure to check the autorecovery file system.

Checking the autorecovery file system

1 **Launch the A/UX Startup utility from the Macintosh OS.**

Double-click the A/UX Startup icon.

2 **Click the Cancel button or press COMMAND-PERIOD.**

3 **Connect the autorecovery partition to a temporary device name.**

Enter the command:

```
pname -cx -s7 "Eschatology 1"
```

where *x* is the SCSI address of the A/UX disk.

4 **Check the consistency of the autorecovery partition.**

Enter the command:

```
fscck /dev/dsk/cxd0s7
```

where *x* is the SCSI address of the A/UX disk.

5 **Verify that the file system is mountable.**

Enter the command:

```
mount /dev/dsk/cxd0s7 /mnt
```

6 **Disconnect the autorecovery partition from its temporary device name.**

Enter the command:

```
pname -u /dev/dsk/cxd0s7
```

7 **Restart the system.**

Choose Restart from the Special menu.

The system will restart. Autorecovery should now be operational.

Boot time errors

The following error can occur while autorecovery is running.

```
filename was not replaceable
```

This means that “replaced” files are missing after autorecovery runs.

When autorecovery determines that a file is invalid, it removes the file and then attempts to replace it with a valid copy from the autorecovery file system. If the autorecovery file system does not contain a valid copy of the file, it will not be replaced. This situation is very unlikely, however, since it means that the CML contains an entry for the file, but the file has never been copied to the autorecovery file system. Alternatively, the file may exist on the autorecovery file system but fails the rules specified in the CML. Consistent use of the `eupdate` and `eu` utilities should prevent this problem.

The autorecovery facility removes invalid files before attempting to replace them by design. Since autorecovery is concerned only with key system files, it removes files identified as invalid, whether or not a suitable replacement is available. You can recover removed files that are not replaced from your *AUX 3.0 CD-ROM* or your backups. See the section “Restoring Missing Files” in Chapter 8.

Run-time errors

The following errors can occur while updating the autorecovery files.

```
Can't lock cml file.
```

```
eu: Can't lock the fcml. Try again later.
```

This message indicates that both `eu` and `escher` are preparing to update the CML at the same time.

The `eu` and `escher` utilities cannot be run at the same time because each must have exclusive access to the CML. Users who receive this message should verify that either `eu` or `escher` is running, but not both. If neither is running, the file `/etc/cml/FCML.lock` may be present. After verifying that no processes are attempting to update the CML, you may remove this lock file.

```
/dev/dsk/cxdysZ previously pnamed  
/dev/rdsk/cxdysZ previously pnamed
```

This means that the `eu` utility will not run. Generally, the device `/dev/dsk/cxdysZ` is still associated with an autorecovery file system.

To remove the association, use the `pname` command by following the steps in the next section.

Clearing autorecovery partition names

1 Check which devices are associated with partitions.

Enter the command:

```
pname
```

The system produces a display something like this:

```
/dev/dsk/c0d0s0: "A/UX Root" "Apple_UNIX_SVR2" [not in ptab file]  
/dev/dsk/c0d0s4: "Eschatology 1" "Apple_UNIX_SVR2" [not in ptab file]  
/dev/dsk/c0d0s31: "Entire Disk" "Apple_UNIX_SVR2" [not in ptab file]
```


2 Disassociate the device and the partition for each device associated with an autorecovery (eschatology) file system.

Enter the command:

```
pname -u device-name
```

For example, in this case, you enter

```
pname -u /dev/dsk/c0d0s4
```

3 Verify that no devices are associated with the autorecovery file system.

Again enter the command:

```
pname
```

The system produces a display something like this:

```
/dev/dsk/c0d0s0: "A/UX Root" "Apple_UNIX_SVR2" [not in ptab file]
/dev/dsk/c0d0s31: "Entire Disk" "Apple_UNIX_SVR2" [not in ptab file]
```

The `escher` utility should now run normally.

Reclaiming disk space

A/UX includes all the files needed to run A/UX locally, as well as many added features. Single-disk systems, however, may not have much room available for user data. You can make more room on your disk in several ways. You can remove software that you don't need to use, such as font files or games. If you're on a network, you can place read-only files, such as the on-line manual pages, on a server and remove them from the client machines. If you are not on a network, you can order printed versions of the *A/UX System Administrator's Reference*, *A/UX Command Reference*, and *A/UX Programmer's Reference* (see the section on conventions in "About This Guide" for a description of each) and remove the man page files from your system. You can also compress infrequently used files to reduce the space required for storing them on the disk, and expand them again on demand. It is also prudent for the system administrator to trim files (for example, log files) that tend to grow over time. To be on the safe side, be sure to make a backup copy of anything you decide to remove.

Removing unneeded files

The `find` command can help you locate and remove unneeded files from your system. Some files, such as those named `core`, are produced by isolated system events. Unless your system is crashing repeatedly, they should be removed.

For example, to remove all the core files, enter the command:

```
find / -name core -exec rm {} \;
```

Large, regular files in the `/dev` directory have usually been created there by accident. To remove regular files larger than about 10K from the `/dev` directory:

1 Locate the files you want to remove.

Enter the command:

```
find /dev -type f -size +20
```

It is usually best to use the `find` command without the `-exec` option first, to verify you will remove the files you *think* you will remove.

2 Verify you actually want to remove all the files located.

If you want to keep any of the files, move them to a directory not in the search path of the `find` command.

3 Remove the files.

Enter the command:

```
find /dev -type f -size +20 -exec rm {} \;
```

4 Replace any files moved in step 2.

Some additional examples of files that may not be needed are:

- `/usr/mail/*`—undelivered mail can be removed
- `/lost+found/*`—unlinked files salvaged by `fsck` can be removed when you have made sure all needed files have been restored
- `/usr/preserve/*`—files from interrupted or damaged `ex` or `vi` sessions
- multiple copies of graphics or text files that can be recreated from a source file

◆ **Note** When removing files, be careful not to remove the parent directory of the files being removed. ◆

Trimming files that grow

As A/UX runs, it creates and appends to assorted log and data files. If no action is taken, these files can eventually grow to a substantial size. Fortunately, it is easy to monitor the growth of these files, and you can automate much of the task of reducing their size. See the section “Automating System Administration with `cron`,” in Chapter 5, for information on automating tasks. Note that some log files are written into only if they exist. Removing such files causes no error condition, but will disable the logging activity.

Other files are produced by the system to log events of administrative interest. These files should normally be trimmed of their earlier entries. For instance, to trim the file `/usr/adm/messages`, enter the commands:

```
tail -20 /usr/adm/messages > mgs.tmp
mv mgs.tmp /usr/adm/messages
```

Some log files have nonreadable records in them but can be trimmed if you know the length of each record (this information is in the man pages). For example, the files `/etc/wtmp` and `/etc/utmp` both have records lengths of 52 characters. These files can be trimmed to some multiple of their record size. To trim the file `/etc/wtmp` to its last twenty entries, enter the commands:

```
tail -1040c /etc/wtmp > w.tmp
mv w.tmp /etc/wtmp
```

Some additional examples of files that may need occasional trimming:

- `/usr/spool/mqueue/syslog`—output from `syslog` daemon
- `/usr/spool/uucp/.LOG`—record of UUCP transactions

Finally, some files must be present, such as log files, but can be truncated to zero length periodically. For instance, to truncate the file `/usr/adm/messages`, enter the command:

```
cp /dev/null /usr/adm/messages
```

Some additional files that can periodically be truncated are:

- `/usr/adm/lpd-errs`—output from `syslog` daemon
- `/usr/spool/lp/l0`—output from `lp`
- `/usr/spool/lp/old1`—output from `lp`

Compressing infrequently used files

The tools available under A/UX for compressing files are `compact`, `compress`, and `pack`. These tools reduce the size of files, saving disk space. The amount of disk storage that you gain from such compression can be substantial, but it takes longer and is more difficult to access the compressed versions than to access noncompressed files.

Only large and infrequently used data files are suitable candidates for compression. For example, in A/UX, the man page files are shipped in compressed form, saving several megabytes of storage. The `man` command has been specially written to deal with compressed files; however, the execution time of the command is longer than that of other commands.

All of these compression tools reduce the size of text files by about 50 percent. Their performance on binary files is weaker, however. The `compress` utility saves about 40 percent, and the other two programs save only about 25 percent of the original storage. Therefore, `compress` should be used when binary or mixed files are involved.

There are also differences in the time the utilities take to run. On text files, `pack` is slightly faster than `compress`. On binary files `compress` is slightly faster. In both cases, however, `compact` takes about ten times as long as the faster of the other two. Clearly, `compact` should be used only when compatibility with another machine requires it. Non-UNIX file-compression utilities can be used; these utilities, however, may not preserve file permissions or ownership, unlike the UNIX utilities described in this section. Unless a compression utility is certified to work with A/UX, it should be used with caution.

To search for files larger than about 100 KB and compress a subset of them, you can use the following procedure:

1 **Locate the files you want to compress.**

Enter the command:

```
find / -type f -size +200 -print
```

2 **Verify you actually want to remove all the files located.**

If you don't want to compress any of the files, back them up or move them to a directory not in the search path of the `find` command.

3 **Compress the files.**

Enter the command:

```
find /dev -type f -size +20 -exec compress {} \;
```

4 **Replace any files moved or backed up in step 2.**

Usage notes

All three compression programs remove their input files during the compression process. The companion decompression programs (`uncompact`, `uncompress`, and `unpack`) do the same. The user should therefore make a copy before compression (or decompression), if the original version is to be retained.

Alternatively, the appropriate “snapshot” decompression program (`ccat`, `zcat`, or `pcat`, respectively) can be used. These tools are analogous to the `cat` command in displaying the uncompressed version of the data on the screen, while leaving the compressed file in place. The `man` command thus uses `pcat` for compressed manual pages, and `cat` for uncompressed ones.

Extensions to names of compressed files

By default, the compression programs append a two-character suffix to the names of their output files. For example, the `compress` utility converts the file `sample` into the compressed file `sample.z`. If the input filename exceeds 12 characters, you get an error message that the file name is too long. This limitation is implemented in order to support SVFS file systems.

Compressing an archive of files

You may sometimes need to compress a collection of small files. You can achieve a substantial saving in storage by first producing a `cpio`, `tar`, or `pax` archive, then compressing the archive. Note, however, that the same archiving utility will be needed after the decompression process when you decompress the archive.

CD-ROM and A/UX

A/UX supports UFS, SVFS, and HFS file systems on CD-ROM drives. Through the A/UX Finder it can also access compact discs recorded with ISO 9660 and High Sierra formats. This lets A/UX users take advantage of the large storage capacity of CD-ROM. A CD-ROM might contain a large number of database files, source code files, or documentation. For instructions on using CD-ROM discs with a local file system, see *Setting Up Accounts and Peripherals for A/UX*. See *Apple CD SC Developer's Guide* for information on creating CD-ROMs.

◆ **Note** Not all manufacturers' CD-ROM drives can use the A/UX SCSI device driver. Use the same drive used to install A/UX, or contact the manufacturer for an appropriate device driver. ◆

You can use a CD-ROM that already contains an A/UX file system. CD-ROM discs that contain UFS or SVFS file systems can be mounted directly into the normal directory hierarchy like a file system on a hard disk, except of course, CD-ROM discs can only be read, not written.

The High Sierra and ISO 9660 formats are supported if you have the driver software located in your System Folder (which may be a personal System Folder). CD-ROM discs having these formats cannot be mounted into the A/UX directory hierarchy. They are accessible from within Macintosh and hybrid applications equipped to interface with them. Instructions for setting up CD-ROM drives can be found in *Setting Up Accounts and Peripherals for A/UX*.

Note that A/UX does not provide support of audio CD-ROM discs; however, the Macintosh OS does provide support for audio CD-ROM discs.

Mounting a CD-ROM as an A/UX file system

If you have a CD-ROM that contains an A/UX file system (UFS or SVFS), you can mount the CD-ROM just like any other read-only file system. For example, if your CD-ROM drive has SCSI address 4, then you can insert a CD-ROM containing an A/UX file system into the drive and mount the first slice by entering

```
mount -r /dev/dsk/c4d0s0 /mnt
```

Use the `-r` option to ensure that the CD-ROM is mounted as a read-only file system; errors will occur if you do not use this option with CD-ROM drives (even if you never try writing to the drive). You can read files on a CD-ROM just as you do those on any other disk. After mounting the CD-ROM, for example, enter

```
ls -R /mnt
```

You might see a listing like this:

```
./mammals:
cats
dogs
zebras
./programs
prog1.c
prog2.c
prog3.c
```

Accessing the man pages on a CD-ROM

By mounting your *A/UX 3.0 CD-ROM* as a read-only file system, you can read the man pages from the compact disc (CD) and remove them from your hard disk, saving roughly 8 MB of disk space. Other directories you may want to access from the release CD-ROM, rather than your hard disk, are:

```
/lib          3 MB
/usr/bin      15 MB
/usr/include  2 MB
/usr/lib      11 MB
```

The following procedure shows you how to set up man page access from a CD-ROM. For the procedure to work, you must be logged in as `root`. Once set up, users obtain man page information from the CD-ROM transparently.

1 Log in as root.

2 Remove the man pages from your hard disk.

Enter the command:

```
rm -rf /usr/catman/*
```

If you later want to reinstall the man pages on your hard disk, you can use Installer or the `cpio` utility to copy them from the *A/UX 3.0 CD-ROM*.

3 Create a mount point for the *A/UX 3.0 CD-ROM*.

Enter the command:

```
mkdir /cdrom
```

4 Set the permissions for the mount point:

Enter the command:

```
chmod 555 /cdrom
```

5 Mount the *A/UX 3.0 CD-ROM*.

Enter the command:

```
mount -r /dev/dsk/c.xd0s0 /cdrom
```

where *x* is the SCSI address of the CD-ROM drive.

6 Verify the files are mounted correctly.

Enter the command:

```
mount
```

The system will display a list of the currently mounted file systems. Here is an example:

```
/dev/dsk/c0d0s0 on / type 4.2 (rw,noquota)
/dev/dsk/c.xd0s0 on /cdrom type 4.2 (ro,noquota)
```


7 Link the new directory to the original directory.

Enter the command:

```
ln -s /cdrom/usr/catman /usr/catman
```

8 Test that the man pages are available.

Enter the commands:

```
su Guest
```

```
man wump
```

The man page for the game `wump` should be displayed.

9 Log out.

7 Managing Printers, Terminals and Modems

Peripheral device drivers / 7-3

Ports / 7-3

Managing the `lpr` print spooler / 7-4

Setting up a terminal / 7-17

Setting up a modem / 7-30

This chapter discusses management of peripheral devices other than mass-storage devices, which are discussed in Chapters 4, 5, and 6. Managing peripheral devices involves connecting devices such as printers, terminals, and modems to your computer; maintaining them; and, when necessary, disconnecting them. There are two aspects to managing a peripheral connection: connecting the hardware to the computer, which in most cases is a rather simple operation, and configuring the software that allows the device and the computer to communicate. This is more involved, but it is not difficult.

To connect a device, refer to the manual that comes with the hardware. This chapter concentrates primarily on the software, the configuration files that you may need to edit so A/UX can properly communicate with the device.

◆ **Note** This chapter discusses using `/dev/tty1` (`/dev/printer`) as a printer port. LocalTalk also uses this port to add your system to the network. You should not plan on using `/dev/tty1` for a printer port, if you plan to attach to a LocalTalk network. ◆

Appendix C, “The System V Print Spooler: `lp`,” provides information on the System V print spooler. The BSD print spooler program, `lpr`, described in the beginning of this chapter, is the preferred program for controlling your print jobs.

If you need instructions for setting up your Apple printer, refer to *Setting Up Accounts and Peripherals for A/UX*.

This chapter discusses connecting another Macintosh as a terminal to your system. You may also connect your system in a network of one or more computers, fully using their capabilities. Networking is explained in *A/UX Networking Essentials* and *A/UX Network System Administration*.

To understand some sections in this chapter, you need to be familiar with the file `/etc/inittab`. This file is discussed in Chapter 2, “System Startup and Shutdown,” and in `inittab(4)` in *A/UX Programmer's Reference*. See also “Changing Run Levels: `init`” in Chapter 1, because use of the `init` command may affect other users.

Peripheral device drivers

All A/UX peripheral devices require a device driver. A **device driver** is the software interface between A/UX and a peripheral device. The peripheral devices that you connect to your machine fall into two categories: those that have device drivers built into A/UX by default, and those that require adding a device driver to the A/UX kernel.

The A/UX kernel has the drivers built in for commonly used Macintosh peripherals, whereas device drivers for less commonly used or non-Apple devices must be added to the kernel before the peripheral is used. It is not readily apparent into which category a particular device falls. In general, if adding the device involves adding an expansion card, such as an Ethernet card, to the computer, you also need to add a device driver.

If you need to add a device driver to your kernel, refer to “The `newconfig` Program” in Chapter 1.

◆ **Note** You do not need to add a device driver to add terminals, printers, modems, tape drives (DAT, 9-track, 1/4”, DCAS, QIC, and Apple Tape Backup 40SC), CD-ROM drives, hard disks, or Apple floppy disk drives to the A/UX system. Support for these devices is already built into the standard A/UX kernel. ◆

Ports

A computer communicates with other equipment through a **port**, which is a physical connection point on your Macintosh. You can attach peripheral devices by connecting them (via cables or connectors) to the ports. Refer to your Macintosh user’s guide for a description of these ports and a diagram that shows their location.

The ports to which you connect peripheral devices such as printers, modems and terminals are called **serial ports**. The Macintosh has two serial ports; they are located on the back of the computer and are marked with a telephone handset icon and a printer icon. Each port takes an 8-pin minicircular connector. Make sure you distinguish the serial port from the Apple Desktop Bus (ADB) ports, which look similar, but are marked with the ADB icon and take 4-pin connectors. The signals on the individual pins of the serial ports are shown in Figure 7-1.

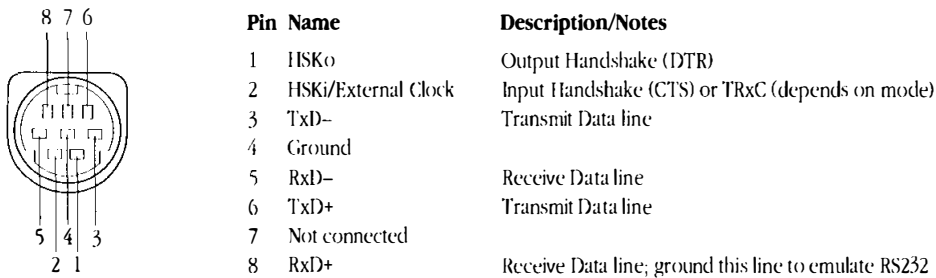


Figure 7-1 Signals on a serial port

The port identified by the phone icon has the A/UX device name `/dev/tty0`; it is also called `/dev/modem`. The port identified by the printer icon has the device name `/dev/tty1`; it is also called `/dev/printer`.

◆ **Note** Even though these ports have the name `/dev/modem` and `/dev/printer`, they are both serial ports. Therefore, you can attach a modem to port `tty1` and a printer to port `tty0`. ◆

Managing the `lpr` print spooler

This section introduces the commands necessary to use the BSD 4.3 `lpr` print spooler system. The system is a collection of programs and files used to manage a printer's operation. This system includes a **print spooler**, which allows more than one person to use the same printer simultaneously or the same user to enter more than one print job in a queue.

The line printer system is composed of two subsystems: the administrative system and the spooler. The administrative system is a series of commands you use to configure and maintain the entire line printer system. The spooler intercepts print requests, schedules them for printing on a specified printer or group of printers, and then selects the appropriate interface for the printer. When configuring the spooler, the system administrator assigns each printer a unique name by which the spooler will identify the printer. The spooler maintains a list of user print requests organized by printer name.

This printer system is controlled and managed through a set of commands that

- queue or cancel printer requests
- query the status of printer requests or of the printer system itself
- prevent or allow queuing printer requests to specific printers
- start or stop the printer system
- change the printer configuration

To use the `lpr` command to print jobs, you should be familiar with the printing instructions in *A/UX Essentials*. This section covers the administrative features of the printer system.

Definitions

Some important terms used in this section are defined as follows:

- A **request** is a print job submitted to the printer using the `lpr` command.
- A **printer** is a unique name by which the the printer system identifies a specific printer.
- A **destination** is a printer. Output is normally routed to the **system default destination** unless the user explicitly requests a particular printer or printer class on the `lpr` command line. See `lpr(1)` in *A/UX Command Reference*.
- A **device** is a piece of hardware such as a printer or modem that can be connected to the computer through a port.
- A **device file** is a file in the `/dev` directory that is associated with a particular device. When the `lpr` system writes to the device file, such as `/dev/tty1`, output is sent to that port. The `lpr` system maintains information necessary to associate each printer with a particular device. By default, the `lpr` system uses an AppleTalk spooler to the printer selected through the Chooser.
- The print spooler's **scheduler**, called `lpd` (line printer daemon), schedules print requests received from the `lpr` command. The `lpd` scheduler runs continuously in the background and is usually started by the `init(1M)` process when A/UX enters multi-user mode. See `lpd(1M)` and `init(1M)` in *A/UX System Administrator's Reference*.

- Each printer is controlled by an **interface program**, which may be shared by more than one printer. Interface programs perform such tasks as setting port speed, selecting printer options, printing header pages, and perhaps filtering certain characters that a particular printer may not know how to handle. The `lpr` system maintains the information necessary to select the proper interface for a given printer.

Setting up the print spooler

In order to be able to spool, or place in a queue, requests to a printer, the printer must be configured within the print spooler system. A/UX is shipped with the AppleTalk and ImageWriter printer queues already created. Therefore, if you are using one of these types of printers, the `lpr` command will spool jobs for you without further configuration. If you have one of these types of printers and have not already set it up for printing, refer to *Setting Up Accounts and Peripherals for A/UX*, as well as the manual for the printer.

To modify your print spooler to add more printers to the spooling system, you must edit the `/etc/printcap` file as well as provide printer filters for any printers other than the LaserWriter or ImageWriter. This includes both Apple and non-Apple printers. See “The `printcap` Database,” next, and “Writing Printer Output Filters,” later in this chapter, for more information on the type of information necessary.

The `printcap` database

The text file `/etc/printcap` contains entries that describe printers used by the print spooler. Each entry is used to describe one printer. You can edit this file using your favorite text editor. As this is an important system file, to save changes to this file you must be logged in as root. As with all important files, you should make a copy of this file before you edit it, in case you find you want to undo your changes.

If you have a printer that does not have an entry in `/etc/printcap`, you may either build a printer definition manually, or ask the printer vendor to supply you with a UNIX `/etc/printcap` entry for that printer.

This section describes some of the printer capabilities that can be defined in the `printcap` file. Refer to `printcap(4)` in *A/UX Programmer's Reference* for a list of all the possible options. Refer to `termcap(4)` in *A/UX Programmer's Reference* for a definition of the format of the file.

The default `printcap` file includes entries for AppleTalk and ImageWriter II printers connected locally via a serial port. You should need to modify this file only if you are using another type of printer.

Printer naming Each entry in the database begins with a list of names separated by a vertical bar that uniquely identifies the printer described by the entry. Any of these names may be used when invoking `lpr` from the CommandShell window. For example, the entry for the ImageWriter II is:

```
iw|iww2|ImageWriter II:\
:lp=/dev/prINTER:br#9600:os#0014001:cs#0004060:fd:\
tr=\f:sd=/usr/spool/lpd/ImageWriter:
```

The default printer in the standard configuration is the generic AppleTalk printer, which is selected using the Chooser. This entry begins with:

```
lp|at|AppleTalk|postscript|Postscript: \
```

The special name `lp` in the name field is used to designate a particular printer entry as the system default destination. (Do not confuse `lp` in the name field with the `lp=/dev/prINTER` definition, which is discussed later.) The environment variable `PRINTER` can be used to override this for a particular user. To change the default printer destination, remove `lp` from the AppleTalk entry and place it as one of the names in the entry of the new default destination.

Printers on serial lines The definitions of the characteristics of the printer follow the printer name(s) in each entry of the database. These definitions are two character variables set in fields separated by colons. Each printer connected directly via a serial communication port must have its characteristics, such as baud rate, defined in the `printcap` entry. An example of some of these definitions for the ImageWriter is as follows:

```
:br#9600:os#0014001:cs#0004060:fd:tr=\f:
```

The `br` entry sets the baud rate for the port, in this case to 9600 baud. The `os` and `cs` entries are less intuitive. For entries such as these, reference `termio(7)` in *A/UX System Administrator's Reference*.

For example, `printcap(4)` indicates that `os` defines the output flag bits which are set for this ImageWriter (that is, the system treatment of output to the printer). The value shown is `0014001`. According to the `termio(7)` manual page, `0014000` for

output flag bits tells the system to expand tabs to spaces, and `0000001` tells the system to postprocess output. Therefore both of these characteristics are in effect when using that ImageWriter.

The `cs` entry sets other control characteristics of the port; in this case, `0004060` tells the system that the connection is a local line (not a dial-up), and the data should be 8-bit.

The `fd` entry doesn't take a value, but its appearance causes the use of hardware-supported flow control or handshaking as described in `termio(7)`.

The `tr` entry indicates that a form-feed (`\f`) should be printed when the queue empties so that the paper can be torn off without taking the printer off line and pressing form feed.

You may also notice backslashes (`\`) at the end of each line of an entry, except at the last line. This backslash tells the computer that the definitions for that entry continue on the next line.

The `lp` entry defines the default port used by `lpr`. If another port were to be used, it would be specified as follows:

```
:lp=/dev/port:
```

where *port* is the name of the port that the printer is connected to (see “Ports” earlier in this chapter). For example, to connect a printer to the modem port you would specify

```
:lp=/dev/tty0:
```

Spooling directory A spooling directory is a directory that holds print requests until the printer is available. Each printer that does not have a separate spooling directory has all its requests sent to the default spooling directory `/usr/spool/lpd`. A print request does not keep track of the specific printer it was sent to after it reaches the spooling directory. Therefore, if several printers have the same spooling directory, a request sent to any one of the printers will be printed on whichever printer is available.

The variable `sd` specifies an alternative spooling directory for a printer. In the following example, `/usr/spool/lpd/ImageWriter` is the specified spooling directory.

```
:sd=/usr/spool/lpd/ImageWriter:
```

Output filters Filters handle device dependencies and perform accounting functions. The output filter definition, `of`, is used when all text data must be passed through a filter before it is sent to the printer. In the following example, all text sent to the AppleTalk printer will first be processed by the filter

```
/usr/spool/lpd/AppleTalk/ofilter.  
:of=/usr/spool/lpd/AppleTalk/ofilter:
```

For more discussion of filters, see “Writing Printer Output Filters” later in this chapter.

Remote printers It is also possible to send your print requests to printers that are connected to other computers. To accomplish this, you would set the `rh` (remote host) and `rp` (remote printer) variables. To set up the `printcap` database for remote printing, see *A/UX Networking Essentials*.

Access control The `rg` variable controls local access to printer queues. For example,

```
:rg:lprgroup
```

requires that a user must be in the group `lprgroup` (which must be defined in `/etc/group`) in order to submit jobs to a particular printer. If the `rg` variable is not defined for a printer, all users have access to that printer. Once the requests are in a printer queue, they can be printed locally or forwarded to another system, depending on the configuration. Remote access is controlled by listing the hosts either in the file `/etc/hosts.equiv` or in `/etc/hosts.lpd`, with one host per line. The `remsh(1)` and `rlogin(1)` commands use `/etc/hosts.equiv` to determine which hosts are specifically designated to allow logins without passwords. The file `/etc/hosts.lpd` controls which hosts have line printer access. For more information on the files and commands involved in remote access, see *A/UX Network System Administration*.

lpr commands

The commands used to administer the line printer system can be divided into two categories: those that any user can use, and those that only a user logged in as root (or a user given permissions to edit `lpr` related files) can use. This section gives a brief description of what each command does. For examples of how these commands are used, see *A/UX Essentials*.

Commands for general use

- `lpr` Usually takes a filename as an argument; submits that file as a print request to the `lpr` system. The request can print on the default system destination or a specified printer or printer class, depending on the options used. See `lpr(1)` in *A/UX Command Reference*.
- `lpq` Shows the line printer queue. This program has two forms of output: the short format (the default), which gives a single line of output per queued job; and the long format, which shows the list of files that make up a job, as well as their sizes. See `lpq(1)` in *A/UX Command Reference*.
- `lprm` Will remove one or several jobs from a printer's spool queue. `lprm` can cancel requests by printer name, request ID number (supplied by `lpq`), or user name. See `lprm(1)` in *A/UX Command Reference*.

Commands for line printer administrators

This section discusses the major commands of the `lpc` program, which provides administrative control over line printer activity. `lpc` may be used to

- disable or enable a printer
- disable or enable a printer's spooling queue
- rearrange the order of the jobs in the spooling queue
- find the status of printers

For additional information on the command format and other `lpc` options, refer to `lpc(4)` in *A/UX Programmer's Reference*. With each of the following commands, you can specify a particular printer, or you can affect all printers controlled by the BSD line printer system.

```
lpc abort printer_name|all
```

The `abort` option stops printing immediately and disables printing for the specified printers. It does not remove any jobs from the queue. Use the `lprm` command to remove jobs.

```
lpc start printer_name|all
```

The `start` option starts printing to the specified printers.

```
lpc disable printer_name|all
```

This command turns spooling off in the specified print queue. When spooling is disabled, `lpr` cannot add more requests to the queue. You may want to turn spooling off while testing new line printer filters, since the special privileges of root allow the user logged in as root to put jobs in the queue even when the queue is disabled. Another use of the `disable` command is to prevent users from putting jobs in the queue when the printer is expected to be unavailable for a long time.

```
lpc enable printer_name|all
```

The `enable` option turns spooling on in the specified print queue and allows `lpr` to put new printer requests in the queue.

```
lpc restart printer_name|all
```

Occasionally some abnormal condition causes the spooling system daemon to quit unexpectedly. If there is no spooling daemon for a printer, print requests in the queue will not print. You may detect this situation with the `lpc status` command. Enter

```
lpc status
```

The result will look like

```
lp:
    queueing is enabled
    printing is enabled
    3 entries in spool area
    no daemon present
```

If you see the line `no daemon present`, use the `lpc restart` command to restart the daemon.

```
lpc stop printer_name|all
```

The `stop` option halts a spooling daemon after the current job completes, which disables printing. This is a clean way to shut down a printer for maintenance. Users can still enter jobs in a spool queue while a printer is stopped.

```
lpc up printer_name|all
```

The `up` option enables everything and, if there are any requests in the printer's queue, starts printing.

```
lpc down printer_name|all
```

The `down` option turns off the spooling queue and disables printing. With the `down` option, you can also enter a message that will appear after an `lpc status` command.

Steps to set-up another printer in the `lpr` system

Follow these steps to add an additional printer to your system. The example is based on another, non-default ImageWriter printer; for other printers, substitute the proper `printcap` definitions.

1 **Log in to your A/UX system as root.**

2 **Copy the `/etc/printcap` file.**

Enter the command

```
cp /etc/printcap /etc/printcap.original
```

This will allow you to restore the original version of the file if something goes wrong while you are editing the file.

3 **Edit the `/etc/printcap` file.**

The best way to accomplish this for another ImageWriter is to copy the entire ImageWriter default entry and make changes for the new printer. For instance, the original ImageWriter entry looks like this:

```
iw|iww2|ImageWriter II:\
:lp=/dev/printer:br#9600:os#0014001:cs#0004060:fd:\
tr=\f:sd=/usr/spool/lpd/ImageWriter:
```

You may want to change the name, the port, and the spooling directory, so the new entry may look like:

```
ix|ix2|newiw| New ImageWriter II:\
:lp=/dev/tty0:br#9600:os#0014001:cs#0004060:fd:\
tr=\f:sd=/usr/spool/lpd/New-ImageWriter:
```

4 **Make the new spooling directory.**

For this example it could be `/usr/spool/lpd/New-ImageWriter`. Use the `mkdir` command. Also, use `chmod` to change the permissions to `775` and `chown` to change the owner and group to `daemon`. When you are finished, `ls -l` should print

```
drwxrwxr-x daemon, daemon /usr/spool/lpd/New-ImageWriter
```

5 **Enable the printer with the `lpc` command.**

The most efficient version of `lpc` for this purpose is

```
lpc up newiw
```

6 **Test the new printer by printing a file.**

```
lpr -Pnewiw /etc/printcap
```

Troubleshooting the `lpr` system

The `lpr` system error messages and possible solutions to problems are explained next. Note that the name *printer* refers to the name of the printer in the `/etc/printcap` database.

`lpr` **error messages**

The following messages are from `lpr`.

```
lpr:printer:unknown printer
```

This means that the printer was not found in the `/etc/printcap` file. Verify that an entry for *printer* in the `/etc/printcap` file is present and correct.

```
lpr:printer:jobs queued, but cannot start daemon
```

This means that the connection to `lpd` on the local machine failed. More than likely the printer server started at boot time has quit or is frozen. Check the local socket `/dev/printer.socket` with the `ls` command to make sure that it still exists. (If it doesn't, no `lpd` process will be running.) As root, enter this command to restart: `lpd: /usr/lib/lpd`

You can also check the state of the master printer daemon with this command:

```
ps -p `cat /usr/spool/lpd.lock`
```

Another possibility is that the `lpr` program is not `setuid` to `root` and `setgid` to the group `daemon`. Check the output of `ls -l /usr/ucb/lpr` looks like

```
-rwsr-sr-x links      root, daemon  size date /usr/ucb/lpr
```

where *link*, *size*, and *date* are defined in `ls(1)`.

```
lpr: printer: printer queue is disabled
```

This message indicates that the queue was turned off by the system administrator with the command

```
lpc disable printer
```

to stop `lpr` from putting files in the queue. The system administrator as `root` can turn the printer back on by using the command

```
lpc enable printer
```

`lpq` **error messages**

The following messages are from `lpq`.

```
waiting for printer to become ready
```

This means that the daemon cannot open *printer*. The most common reason for this is that the printer is off line. The message may also be displayed if the printer is out of paper, or the paper is jammed. The actual reason depends on the meaning of error codes returned by the device driver. Check that the printer is ready for printing. If the printer is okay, try the `lpc restart` command. If all else fails, reboot the system.

```
printer is ready and printing
```

This means that the `lpq` program checks to see if a daemon process exists for *printer* and prints the file status located in the spooling directory. If the daemon is not working, `root` can use `lpc` to stop the current daemon and start a new one.

waiting for *host* to come up

This message implies that a daemon is trying to connect to the remote machine named *host* to send the files in the local queue. If the remote machine is up, `lpd` on the remote machine has probably terminated or has hung and should be restarted.

sending to *host*

This means that the files should be in the process of being transferred to the remote *host*. If they are not, the local daemon should be terminated and started with `lpc`.

Warning: *printer* is down

This means that the selected printer has been marked as unavailable with the `lpc down` command. After figuring out why the printer was brought down, use the `lpc up` command to bring the printer back up.

Warning: no daemon present

This means that the `lpd` process overseeing the spooling queue, as specified in the `lock` file in that directory, does not exist. This normally occurs only when the daemon has unexpectedly quit. The error log file for the printer and the `syslogd` logs should be checked for a diagnostic from the former process. To restart a line printer daemon (`lpd`), enter

```
lpc restart printer
```

`lprm` **error messages**

The following message is from `lprm`.

```
lprm: printer:cannot restart printer daemon
```

This message is the same as when `lpr` prints that the daemon cannot be started. This normally occurs only when the daemon has unexpectedly quit. The error log file for the printer and the `syslogd` logs should be checked for a diagnostic message.

lpd error messages

The `lpd` program logs every message using the `syslog` file. Most messages logged by the `lpd` program relate to files that cannot be opened and usually mean that the `printcap` file or the permissions of the files are incorrect. Files may also be inaccessible if users bypass the `lpr` program when printing.

lpc error messages

The following messages are from `lpc`.

```
couldn't start printer
```

This is the same as when `lpr` reports that the daemon cannot be started. This normally occurs only when the daemon has unexpectedly quit. The error log file for the printer and the `syslogd` logs should be checked for a diagnostic.

```
cannot examine spool directory
```

Error messages that begin with “cannot” usually result from incorrect ownership or protection mode of the lock file, spooling directory, or `lpc` program.

Writing printer output filters

Filters supplied with A/UX handle printing and accounting for AppleTalk and ImageWriter printers. For other devices or accounting methods, you may have to create a new filter.

Writing filters requires some understanding of programming. To see an example of a filter provided with A/UX, look at one of the filter programs in the `/usr/spool/lpd/AppleTalk` directory. The following information is provided for the more advanced system administrator.

Filters are spawned by `lpd` with their standard input being the data to be printed and their standard output the printer. Standard error is attached to the `lf` file for logging errors, or `syslogd` may be used for logging errors. A filter must return one of these exit codes, depending on the circumstance:

- 0 If there were no errors
- 1 If the job should be reprinted
- 2 If the job should be thrown away

When `lprm` sends a terminate signal to the `lpd` process controlling printing, it sends a `SIGINT` signal to all filters and descendants of filters. This signal can be trapped by filters that need to do cleanup operations, such as deleting temporary files.

Arguments passed to a filter depend on the type of the filter. The `of` filter is called with the following arguments:

```
filter -width -length
```

The *width* and *length* values come from the `pw` and `pl` entries in the `printcap` database. The `if` filter is passed the following parameters:

```
filter [-c] -width -length -iindent -n login -h host accounting-file
```

The `-c` flag is optional, and only supplied when control characters are to be passed uninterpreted to the printer (when using the `-l` option of `lpr` to print the file). The `-w` and `-l` parameters are the same as for the `of` filter. The `-n` and `-h` parameters specify the login name and host name of the job owner. The last argument is the name of the accounting file from `printcap`.

All other filters are called with the following arguments:

```
filter -xwidth -ylength -n login -h host accounting-file
```

The `-x` and `-y` options specify the horizontal and vertical page size in pixels (from the `px` and `py` entries in the `printcap` file). The rest of the arguments are the same as for the `if` filter.

Setting up a terminal

Under A/UX, you can add additional terminals to your Macintosh by connecting them to the serial ports on the back of your system. While these additional terminals cannot run the Finder, they can interact with A/UX in command line mode—the mode used with `CommandShell`.

For each serial port that has a terminal attached, the Macintosh must have port initialization instructions. The computer uses the port initialization instructions found in the `/etc/inittab` file and the port definitions found in the `/etc/gettydefs` file.

The `inittab` file

The first step in adding a terminal is to initiate a `getty` on the port that connects to the terminal. A `getty` is a process that puts the `login` prompt on the screen. This is done in the `inittab` file.

Three of the lines in the file should look like these

```
co::respawn:/etc/loginrc          # Console port
00:2:off:/etc/getty tty0 at_9600  # Port 0 (modem); set to "respawn"
01:2:off:/etc/getty tty1 at_9600  #Port 1 (print.); set to "respawn"
```

Details on configuring `inittab` for attaching a terminal device to your Macintosh are described later in this chapter. For now, note which lines refer to `/dev/tty0` (modem) and `/dev/tty1` (printer).

Format of `/etc/inittab`

Each line in `/etc/inittab` is an entry containing four fields separated by colons; an entry can be followed by an number sign (#) and a comment. The format of each entry is *id:run-level:action:command # comment*

For example, the following line in the `/etc/inittab` file is distributed with the standard system:

```
co::respawn:/etc/loginrc  #spawn Login or getty for console
```

The *id* field is an arbitrary identifier of one to four characters that makes the entry unique. Although the identifier is arbitrary, convention dictates that the identifier in some way represents which port the entry refers to. For example, in the *id* field of the above line, `co` refers to the console port; `01` is usually the identifier for `/dev/tty01`.

The *run-level* field tells `init` whether to process the entry. The *run-level* field can be any number from 0 to 6 (2, for multi-user, is the most common). If the *run-level* field is empty, `init` processes the entry at all run levels.

The *action* field specifies how to execute the command field if the entry's *run-level* field matches, or is less than, the system's current run level. Possible *action* values are:

<code>respawn</code>	If you specify <code>respawn</code> as the <i>action</i> field, the command runs whenever the run level matches the run level of the entry. If you leave the <i>run-level</i> field empty, the process runs at all times and all run levels. Processes with an action of <code>respawn</code> , when killed, will immediately regenerate themselves. In the case of the <code>getty</code> command, where this is most often used, this is the desired behavior. If you want to disable a process whose action is <code>respawn</code> , you must edit the entry in the <code>inittab</code> file and change the action field to <code>off</code> before killing the process. For more information on killing a process, see <code>kill(1)</code> .
<code>off</code>	Tells the system to turn off this command for all run levels.
<code>once</code>	Tells the system to turn on this command for this run level. If the command terminates for any reason, it will not regenerate itself; no further action is taken.
<code>wait</code>	Tells the system to wait until the current command is completed before <code>init</code> goes to the next line. The <code>wait</code> parameter is very important when you mount devices or file systems that require the current command to be completed before the next is begun.

The fourth field is the command to be executed. When the run level of an entry is less than or equal to the current run level, the command named in the fourth field is executed.

The comment, preceded by a number sign (`#`), indicates that the line refers to the console port. Any text after the number sign is ignored by `init`.

The `gettydefs` file

Another file that is important in managing peripheral devices is `/etc/gettydefs`. The `gettydefs` file contains information used by the `/etc/getty` program (the process that waits for a user to log in). This file is composed of individual entries, each with five fields separated by a number sign (`#`). Each entry is separated from the others by a blank line. Except for the *prompt* field, you may insert space (blanks or tabs) between the fields for readability. Entries in `/etc/gettydefs` look similar to the following:

```
co_9600 # B9600 # B9600 SANE TAB3 # ~MODEM ~DTR ~FLOW #
\r\n\nApple Computer Inc. A/UX\r\n\nlogin: # co_4800
```

◆ **Note** In this example, output lines are wrapped onto two lines. When a line in the file has more characters than will fit on a single terminal line, the line will wrap onto the next screen line even though there is only one corresponding file line. ◆

The entries are of the form

label # *initial-flags* # *final-flags* # *flow-control* # *prompt* # *next-label*

where the fields are interpreted as follows:

<i>label</i>	The string that <code>getty</code> tries to match so that it can use the entry. If the second argument to <code>/etc/getty</code> in an <code>inittab</code> entry is, for example, <code>co_9600</code> , then the entry in <code>gettydefs</code> that starts with <code>co_9600</code> is used.
<i>initial-flags</i>	The settings used on the terminal before the <code>login</code> program is executed. The only critical flag at this point is the <code>B</code> flag, which is used to decide the communications baud (line speed). In the example above, the flag is set to 9600, but it could be any valid baud rate.
<i>final-flags</i>	The settings used on the terminal after <code>login</code> is executed. Again, speed (for instance, <code>B9600</code>) is critical. <code>SANE</code> is a composite setting; it takes care of other important terminal settings without your having to set them individually. <code>TAB3</code> specifies that tabs will be sent to the terminal as spaces. <code>HUPCL</code> specifies that the software should hang up the line if the hardware connection is closed. For more information on these flags, see <code>termio(7)</code> in <i>A/UX System Administrator's Reference</i> and <code>gettydefs(4)</code> in <i>A/UX Programmer's Reference</i> .
<i>flow-control</i>	Specifies the type of flow control to be used on the line. The settings can be <code>APPLE</code> , <code>DTR</code> , <code>MODEM</code> , and <code>FLOW</code> . Here also you can subtract a setting by prefixing it with a tilde (~).
<i>prompt</i>	The login prompt that appears on the terminals.
<i>next-label</i>	Another entry in the <code>/etc/gettydefs</code> file for <code>getty</code> to try in case the current entry causes a failure. If <code>getty</code> cannot read the keyboard input using the current definitions, it tries the entry denoted in this field. For instance, if the user logs in at a terminal set to communicate at 4800 baud, but the <code>getty</code> on the port refers to an entry whose <i>initial-flags</i> are set to 9600 baud, the <code>getty</code> will not accept or be able to understand the input. When this happens, <code>getty</code> looks at this field for an alternative entry to try.

In the example used from `/etc/inittab`,

```
00:2:off:/etc/getty tty0 at_9600 # Port 0 (modem); set to  
"respawn"
```

`/etc/getty` is in the *command* field with two arguments. The first, `tty0`, specifies the port on which `getty` should run. The second, `at_9600`, refers to an entry in the `/etc/gettydefs` file with the *label* `at_9600`. The characteristics specified in the entry beginning with `at_9600` are used to set up communications on the port `/dev/tty0`.

Using another Macintosh computer as a terminal

This section discusses how to set up another Macintosh computer for use as a terminal on your system. In this section, the command line interface is used to edit the `inittab` file. See *Setting Up Accounts and Peripherals for A/UX* to learn about the `setport` Commando dialog box that simplifies this procedure. (The command line interface use of the `setport` command is discussed later in this chapter in “Setting Up a Serial Port: `setport`.”)

It is possible to connect another computer to a system through a serial line, just as though it were a terminal. For example, you can treat a Macintosh computer running the MacTerminal application exactly like a terminal. As long as the files `inittab` and `gettydefs` are configured to allow logins on the appropriate port, successful communication can take place, even though the system has no way of knowing that it is communicating with a computer and not merely a terminal.

There are numerous advantages to replacing a terminal with a personal computer that emulates a terminal. These advantages include the ability to scroll back through output and to transfer files between the computers.

Attaching a Macintosh computer as a terminal

Attaching a terminal to your system allows a second user to access A/UX in console emulator mode while you or someone else is logged in at the console. This section describes how to attach a Macintosh Plus, running a terminal-emulator application such as MacTerminal, to your Macintosh.

◆ **Note** These instructions also apply to a Macintosh SE, although references are to the Macintosh Plus. ◆

To connect a Macintosh Plus computer as a terminal, you need an Apple system cable, such as a Macintosh Plus to ImageWriter II cable (part 590-0552 or M0187), with 8-pin minicircular connectors at both ends.

Important Be sure that the power for both the computer you are using as a terminal and your A/UX system is turned off before beginning this procedure.

1 Plug one end of the cable's circular connector into the modem port on the back of your A/UX system.

The modem port is identified by the symbol of a telephone handset.

2 Connect the free end of the cable to the modem port on the Macintosh Plus.

The modem port is located on the back of the Macintosh Plus and is identified by the same symbol as the modem port on your A/UX system.

3 Turn on the Macintosh Plus and configure your terminal-emulator application.

Start your terminal-emulator application on the Macintosh Plus. Consult the user's guide for your application to set the terminal characteristics shown below. (If you are using MacTerminal, for example, you configure the application by selecting Terminal and Compatibility from the Settings menu.)

- terminal type: VT100™ or VT102 (The VT100 and VT102 are popular terminals manufactured by Digital Equipment Corporation. They are emulated by nearly all communications programs. The VT100 is the terminal A/UX expects by default to find at `/dev/tty0`. See `ttytype(4)` in *A/UX Programmer's Reference* for more information about how A/UX is configured for default terminals.)
- line width: 80 columns
- mode: ANSI
- baud: 9600

- bits per character: 8
- parity: none
- connection port: modem
- connection: to another computer (that is, instead of to a modem)
- handshake: XON/XOFF

4 Turn on your A/UX system, and log in to A/UX as root.

◆ **Note** It is a good idea to make a copy of the `/etc/inittab` file before you make any changes. When you change an important system file like this, it is always a good idea to save a copy in case you make a mistake. You can then copy the old file back over the changed version and return your system to its previous state. ◆

5 Copy the `/etc/inittab` file.

Change to the `/etc` directory and enter

```
cp inittab inittab.old
```

6 Edit the `/etc/inittab` file.

Each of the entries shown describes a separate port. The modem port is `tty0`, find the line that looks like

```
00:2:off:/etc/getty tty0 at_9600 # Port 0 (modem); set to  
"respawn"
```

This is the line in `inittab` that governs `tty0`. This is apparent both by the `tty0` argument to `/etc/getty`, and by the comments depicting `Port 0`.

Remember from Chapter 2, "System Startup and Shutdown," that the lines are divided into fields, separated by colons, with the form

id: run-level: action: command

The *action* field in `/etc/inittab` entries for terminals to become active should be the word `respawn`. Change the word `off` to `respawn`. The new line looks like

```
00:2:respawn:/etc/getty tty0 at_9600 #Port 0 ...
```

Save the `/etc/inittab` file.

7 Enter the command `init q`

This command will reread `inittab` and change all processes accordingly. In this case, a `getty` will be initialized on the port whose *action* field was changed to `respawn`.

◆ **Note** After entering this command, you should immediately see the root command prompt again. If the system returns any additional messages, you've entered the command incorrectly. In that case, reenter the command. ◆

8 Verify that the `getty` process is running.

Enter the command

```
ps -ef | grep getty
```

A line similar to the one below should appear on your screen:

```
root 82 1 0 11:43:37 0 0:01 /etc/getty tty0 at_9600
```

The numbers in your output will be different, but the line should mention `tty0`, which shows that a `getty` process has successfully spawned at `/dev/tty0`—the modem port. This is the process that will allow a login on the terminal.

If you receive no output from the command, or if `tty0` is not one of the `getty` processes that appears from the `ps` command, enter the following command:

```
cp /etc/inittab.old /etc/inittab
```

After entering this command, begin these instructions again at step 4.

9 **Log in to A/UX from the Macintosh computer you are using as a terminal.**

You should now have a login prompt on the other Macintosh. Log in as any valid user to test the connection. Enter your password when prompted, and press RETURN to accept the VT100 terminal type. Your Macintosh computer is now serving as a functioning terminal for A/UX.

If you don't get a login prompt on the other Macintosh, check the cable connection or your settings on the terminal-emulation application. If the command in step 8 was successful, the problem is not likely to be related to A/UX.

Attaching a VT100, VT102, or other terminal

You can attach a VT100, VT102, VT100 emulator, VT102 emulator, or other terminal as a peripheral device in much the same way as using a Macintosh computer as a terminal. (See previous section "Attaching a Macintosh Computer as a Terminal.")

Important Be sure that the power for both the terminal and your A/UX system is turned off before beginning this procedure.

1 **Plug the A/UX end of the cable into the modem port on the back of your A/UX system.**

2 **Connect the free end of the cable to the terminal.**

See the terminal's manual for details on the terminal's connectors.

3 Turn on the terminal and configure the terminal settings to be compatible with A/UX.

Consult the terminal's manual to set the terminal characteristics shown below.

- terminal type: VT100 or VT102, if available. The VT100 is the terminal A/UX expects by default to find at `/dev/tty0`. If your terminal is not a VT100 or able to emulate a VT100, you must edit the `/etc/ttytype` file. See `ttytype(4)` and `stty(1)` for more information about how A/UX is configured for default terminals.
- line width: 80 columns
- mode: ANSI
- baud: 9600
- bits per character: 8
- parity: none
- handshake: XON/XOFF

4 To attach non-VT100 terminals, edit the `/etc/ttytype` file.

Replace VT100 in the following line:

```
VT100    tty0
```

with the designation of the terminal, for example WYSE350, or VT102.

To determine the name of the terminal, check the `/etc/termcap` file. If your terminal does not exist as an entry (see `termcap(4)`), consult your terminal's manual or manufacturer for an entry in the file that matches yours or that you can modify. Use this designation for `tty0`. (Information about setting your terminal's parity bits and baud rates can be supplied by the retailer or manufacturer.)

5 Turn on your A/UX system, and log in to A/UX on your system as root.

- ◆ **Note** It is a good idea to make a copy of the `/etc/inittab` file before you make any changes. When you change an important system file like this, it is always a good idea to save a copy in case you make a mistake. You can then copy the old file back over the changed version and return your system to its previous state. ◆

6 Copy the `/etc/inittab` file.

Change to the `/etc` directory and enter

```
cp inittab inittab.old
```

7 Edit the `/etc/inittab` file.

Each of the entries shown describes a separate port. The modem port is `tty0`, find the line that looks like

```
00:2:off:/etc/getty tty0 at_9600 # Port 0 (modem); set to  
"respawn"
```

This is the line in `/etc/inittab` that governs `tty0`. This is apparent both by the `tty0` argument to `/etc/getty`, and by the comments depicting `Port 0`.

As explained in Chapter 2, “System Startup and Shutdown,” the lines are divided into fields and separated by colons, with the form

id:run-level:action:command

The *action* field in `/etc/inittab` entries for terminals to become active should be the word `respawn`. Change the word `off` to `respawn`. The new line looks like

```
00:2:respawn:/etc/getty tty0 at_9600 # Port 0 ...
```

Save the `/etc/inittab` file.

8 Enter the command `init q`

This command will initialize a `getty` process on the port whose *action* field was changed to `respawn`.

◆ **Note** After entering this command, you should immediately see the root command prompt again. If the system returns any additional messages, you’ve entered the command incorrectly. In that case, reenter the command. ◆

9 **Verify that the `getty` process is running.**

```
ps -ef | grep getty
```

A line similar to the one below should appear on your screen:

```
root 82 1 0 11:43:37 0 0:01 /etc/getty tty0 at_9600
```

The numbers in your output will be different, but the line should mention `tty0`, which shows that a `getty` process has successfully spawned at `/dev/tty0`—the modem port. This is the process that will allow a login on the terminal.

If you receive no output from the command, or if `tty0` is not one of the `getty` processes that appears from the `ps` command, enter the following command:

```
cp /etc/inittab.old /etc/inittab
```

After entering this command, begin these instructions again at step 4.

10 **Log in to A/UX from the terminal.**

You should now have a login prompt on the terminal. Log in as `root` or any other valid user to test the connection. Enter your password when prompted. Also enter, when prompted, the correct terminal type. You now should have another functioning terminal for A/UX.

If you don't get a login prompt on the terminal, check the cable connection or your settings on the terminal. If the command in step 9 was successful, the problem is not likely to be related to A/UX.

Setting up a serial port: `setport`

The `setport` command provides a shortcut to add or modify serial ports in the `inittab` file. To use the `setport` Commando dialog box for setting up modems and terminals, see *Setting Up Accounts and Peripherals for A/UX*. This section describes how to use the `setport(1M)` command on the A/UX command line.

The syntax for the `setport` command is

```
setport -r [-s speed] device-file
```

```
setport -o [-s speed] device-file
```

The command's options and arguments are as follows:

- `-r` Sets the port to permit login sessions (The *action* field is `respawn`)
- `-o` Sets the port to disallow login sessions (The *action* field is `off`)
- `-s speed` Specifies the initial device speed. If this option is not included, the speed is set to 9600 baud. For modems, 1200 or 2400 is usually correct

device-file The name of an existing serial port in the `/dev` directory, such as `/dev/tty0`. The `setport` command creates an entry in `/etc/inittab` for *device-file*, if necessary

For example,

```
setport -r -s 2400 tty0
```

enables a login session on the serial port `/dev/tty0` with the initial speed set to 2400. Or

```
setport -r tty0
```

would accomplish the same as Steps 7 and 8 in the previous section "Attaching a VT100, VT102 or Other Terminal."

Note that `setport` requires that *device-file* already exists in the `/dev` directory. For `/dev/tty0` and `/dev/tty1`, you do not need to do anything as the system arrives with these device files already configured. If you add expansion cards to your system, you may have to make these device files or you may receive installation scripts which will make them for you. Device files are special files that A/UX uses to connect between the software and an actual physical port. For example, the device file for serial port 0 is `/dev/tty0`. For more information about device files, see *A/UX System Administrator's Reference* and the `mknod(1M)` manual page.

Setting up a modem

This section describes how to configure a modem using the A/UX command line interface. See *Setting Up Accounts and Peripherals for A/UX* for a discussion of the `setport` Commando dialog box that simplifies this procedure. Also covered in this section is a brief discussion of originating an outgoing call using the `cu` command. (See “Dialing Out on your Modem” later in this chapter.) For more information on your communication software options, see *Setting Up Accounts and Peripherals* and *A/UX Network System Administration*.

A modem can function in incoming or outgoing mode. If you have a modem plugged in, outgoing calls always work. Incoming calls work if there is a `getty` configured on the line in `/etc/inittab`. If desired, you can set up a dial-out modem on one serial port and a dial-in modem on another port. This section uses the example of a modem on port `tty0`, but this is arbitrary. You can put a modem on any serial port.

- ◆ **Note** Although the same modem may be configured for both incoming and outgoing calls, you cannot have incoming and outgoing calls simultaneously on the same modem. ◆

Dial-out access only

This section describes how to configure a modem on port `tty0` to work as an outgoing modem.

Important Be sure that the power for both the computer and the modem is turned off before beginning this procedure.

- 1 **Plug the A/UX end of the modem cable into the modem port on the back of your A/UX system.**

2 **Connect the free end of the cable to the modem.**

See the modem's manual for details on the connection.

3 **Configure the modem's settings to be compatible with A/UX. Turn on the modem.**

Consult the modem's manual to set the modem's characteristics.

4 **Turn on your A/UX system, and log in to A/UX on your system as root.**

If the root command prompt appears on the console, you're already logged in as root.

◆ **Note** It is a good idea to make a copy of the `/etc/inittab` file before you make any changes. When you change an important system file like this, it is always a good idea to save a copy in case you make a mistake. You can then copy the old file back over the changed version and return your system to its previous state. ◆

5 **Copy the `/etc/inittab` file.**

Change to the `/etc` directory and enter

```
cp inittab inittab.old
```

6 **Edit the `/etc/inittab` file.**

Each of the entries in this file describe a separate port. The modem port is `tty0`, find the line that begins with `00`, it will look similar to

```
00:2:respawn:/etc/getty tty0 at_9600 # Port 0 (modem); set  
to "respawn"
```

Change the line to make it look like

```
00:2:off:/etc/getty tty0 at_9600 # Port 0 ...
```

Save the `/etc/inittab` file.

7 **Enter the command** `init q`

This command will remove any `getty` processes on port `tty0`, allowing outgoing calls on that port.

8 **Verify that the** `getty` **process is not running.**

```
ps -ef | grep getty
```

If you receive no output from the command, or if `tty0` is not one of the `getty` processes that appear from the `ps` command, you have successfully killed the `getty`. Whenever the system is in multi-user mode, the new entry in `inittab` will be effective, and your modem will function in outgoing mode.

Dial-in access

You may want to allow other people or other computers to use your system via dial-in access. Once your system is set up to receive calls, anyone dialing in to your system can use it as though he or she were connected directly to your computer via a terminal. Remember, you may also use the `setport` command to make this process easier. This section describes how to configure your system with a modem allowing dial-in access.

Before beginning this section, follow Steps 1 through 4 in the previous section “Dial-out Access Only,” to connect the modem to your system.

1 **Copy the** `/etc/inittab` **file.**

Change to the `/etc` directory and enter

```
cp inittab inittab.old
```

2 **Edit the `/etc/inittab` file.**

To make your modem work as an incoming device on `tty0`, find the line in `/etc/inittab` that governs `tty0`, it will look like:

```
00:2:off:/etc/getty tty0 at_9600 # Port tty0
```

and make sure the word in the third field is `respawn`, and the `at_9600` is changed to `mo_1200`. The new line will look like:

```
00:2:respawn:/etc/getty tty0 mo_9600 # Port tty0 dialup
```

If the baud rate of your modem is 2400, change `mo_1200` to `mo_2400`.

This line now tells the system to spawn a `getty` on port `tty0` at run level 2. This allows the modem on port `tty0` to function as an incoming device only.

◆ **Note** For dial-in lines on an Apple Personal Modem, `/etc/apm_getty` needs to run instead of `/etc/getty`. The `/etc/apm_getty` program sends the control sequences that enable the modem to auto-answer and then executes the `/etc/getty` program. ◆

The Apple Personal Modem line would look like

```
00:2:respawn:/etc/apm_getty tty0 mo_1200 # Port tty0 dialup
```

3 **Enter the command `init q`**

The new entry in `/etc/inittab` takes effect and your modem functions in incoming mode. You may also have to instruct your modem to auto-answer; consult your modem manual for details.

Dialing out on your modem

First, you must learn the phone number of a modem attached to another computer that is set up to receive calls. Once you have this number, you can use different commands to get your modem to talk to the other computer. The `cu` command is the one used in this manual for testing purposes.

1 Call out on the modem.

Most modems operate at 300, 1200, or 2400 baud (or higher). Using a modem at 300 baud can be extremely slow. If possible, operate the modem at 1200 or 2400 baud. You can specify the baud rate in the `cu` command by using the `cu` flag option `-s`:

```
cu -sspeed
```

For example,

```
cu -s1200
```

Another option to `cu` is the `-l` option. The `-l` option stands for “line” and tells `cu` which port the modem is on. Combining these two options gives us the command line that allows you to dial out:

```
cu -s1200 -ltty
```

Enter this command line from a CommandShell window. After a moment (you need to wait for the modem to connect the line), the word `Connected` should appear on the screen. This means you are connected to the modem. If you have any problem at this stage or later, check that the ownership of `/dev/tty0` is the same as the ownership of `/usr/bin/cu`.

Hayes-compatible modems

Each modem has a specific way of communicating and doing what you want once you are connected to it. If your modem is not Hayes compatible, such as an Apple Personal Modem, your modem manual will have information on how to use the modem to communicate.

Enter the command that tells your modem to dial a phone number. If your modem is Hayes compatible, enter:

```
ATDT phone-number
```

The *phone-number* field must be the number of a computer that is ready to receive a call. If you are in an office and must request an external line by dialing a number, say 9, before the phone number, then the command to try is

```
ATDT 9, phone-number
```

This tells the modem to dial 9, wait, and then dial the phone number.

Once you are connected to the other computer, that computer's login prompt appears on your screen. You can now log in and treat this remote computer as if you were sitting at a terminal directly connected to it. See `cu(1C)` in *A/UX Command Reference*.

2 **End the login session on the other computer.**

Use the `logout` command, or enter CONTROL-D.

3 **Wait one second, then enter `+++` and wait another second.**

You will see the message `OK` on your screen.

4 **Enter `ATH`**

This tells your modem to hang up. The modem responds by having your computer screen display the message `OK`. If your modem is not Hayes compatible, the manual that comes with it describes how to disconnect it from a remote computer.

5 **Enter the two-character sequence `TILDE-PERIOD (~.)`**

This terminates the session with `cu`. The word `Disconnected` appears.

8 Checking the A/UX File System: `fsck`

Introduction to `fsck` / 8-3

Overview of the A/UX file system / 8-3

How `fsck` works / 8-13

Using `fsck` / 8-17

`fsck` error messages / 8-22

Restoring missing files / 8-58

This chapter first describes the structure of the two UNIX types of file systems supported by A/UX:

- UFS—the Berkeley File System (also known as 4.2)—which is the file system on the root partition
- SVFS—the System V File System (also known as 5.2)—which is provided for compatibility with current installations

Note that this information does not apply to the Macintosh file system. This chapter discusses how `fsck` works and how to use it.

If you are currently trying to correct errors with `fsck`, turn to the section “`fsck` Error Messages.” This section contains the suggested responses for various problems. Alternatively, you can consult the index of this guide to locate error messages.

In many cases, `fsck` can fix damage to a file system. Sometimes, however, `fsck` can report only cryptic messages about the damage that has been done. In these cases, there are generally two solutions: a system administrator can partially repair the file system with the help of `fsck` and restore any missing files from backups, or a technical expert who knows the structure and functioning of the file system can resolve the problem by using a debugging tool, `fsdb`. The goal of this chapter is to provide you with enough information to use the first of these solutions.

Introduction to `fsck`

The file-system check program `fsck` locates and resolves inconsistencies within a file system. It is part of the normal startup sequence, as described in Chapter 2, “System Startup and Shutdown.” In the standard A/UX distribution, `fsck` is run automatically on the root file system and file systems listed in `/etc/fstab`. If `fsck` detects errors during the automatic startup procedures, a dialog box is presented asking if you want to repair the errors. If you do not click the Repair button, or if a message that says the file system cannot be repaired is displayed, you must run `fsck` through the command line interface before you can mount the file system. The section “Using `fsck`,” later in this chapter, tells you how to do this.

- ◆ **Note** File system problems do not go away with time; instead, they only get worse. Run `fsck` any time that you suspect inconsistencies within the file system, such as immediately following a power failure or any system crash. ◆

Overview of the A/UX file system

The A/UX operating system treats almost everything in its environment as a file. To operate on a file in the A/UX environment, you need refer to it only by name. The general functions of the A/UX file system are to

- support the seemingly simple interface on A/UX mass-storage media (hard disks, floppy disks, CD-ROM drives, and tape cartridges)
- permit the kernel to find data on the disk
- load the data into main memory
- periodically update the disk with the modifications performed on the data in main memory

Sometimes this updating fails, usually because of a power failure or improper system shutdown; this may result in inconsistencies within the file system. In most cases you can resolve these inconsistencies by using the `fsck` program.

The `fsck` program checks the location of files on disk and uses redundancies and known parameters to resolve inconsistencies. A **redundancy** is information that the system maintains in more than one place, such as the size of each file and the number of blocks not currently in use. A **known parameter** is information about the file system that does not change, such as the number of characters per block or the number of blocks in a disk.

It is helpful to understand the organization of the A/UX file system and some of the commands that manipulate this organization before you begin to work with `fsck`. You may wish to review the information in the section “File Systems,” in Chapter 4. This section gives a brief overview of the relevant file and directory information.

Blocks and bytes

A file system is divided into units called *blocks*. A block on the disk is called a **physical block** and is a contiguous sequence of bytes (usually 512 bytes in length). To speed up the disk I/O operations, the A/UX file system works with more than one physical block at a time; this entity is called a **logical block**. This allows the operating system to utilize more data per disk access. The number of physical blocks per logical block is file-system dependent. SVFS typically uses two physical blocks per logical block—this is referred to as a 1-kilobyte file system. In comparison, UFS typically uses 4K or 8K file systems (8 to 16 physical blocks per logical block), which increases disk performance.

Inodes

Inodes contain information about files, the most important of which is a list of logical blocks where a file’s contents are to be found. The inode does not point to a single location on the disk, but to several discrete locations (see the next section, “Direct and Indirect Blocks”).

Unlike files and directory files, inodes are of a fixed size and reside in fixed locations on the disk. For this reason, inodes have i-numbers instead of names. The i-number 30 points to the 30th inode in the inode area on the disk. Directory files are simply lists correlating inodes and file names. To determine a file’s inode from its name you can use the command

`ls -i filename`

To determine a file's name from its inode you can use the command

`ncheck -i inode /dev/rdisk/cxxd0sz`

The value of *x* is the SCSI ID of the hard disk, the value of *z* is the slice number associated with a particular disk partition.

Figure 8-1 illustrates the relationship between the directory `/users/demo`, a file in a directory named `letter`, the i-number and inode associated with `letter`, and the disk locations where the contents of `letter` are stored.

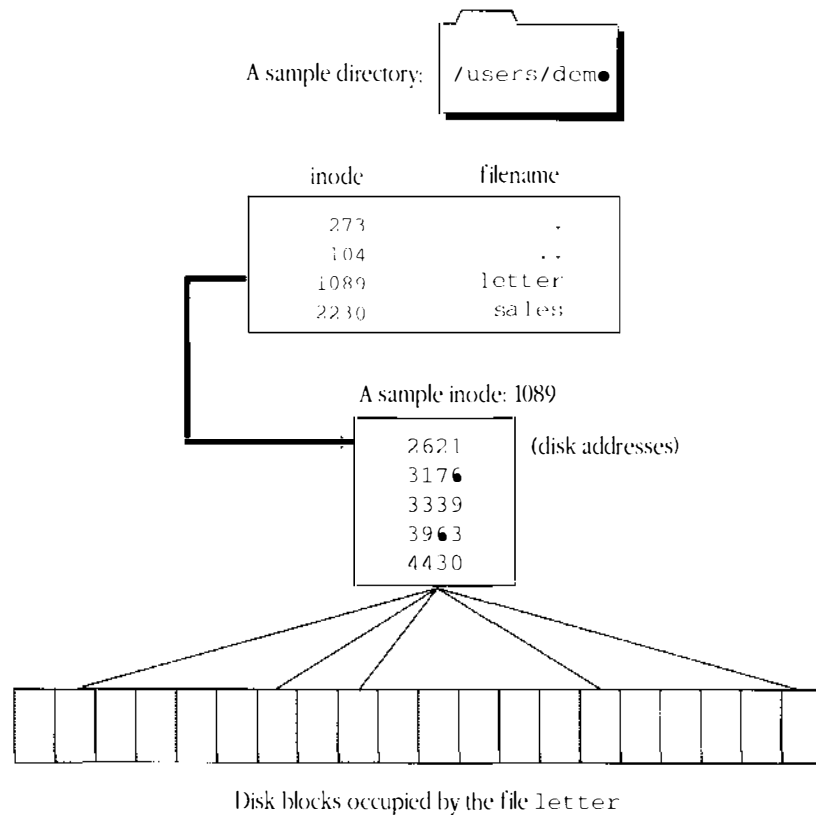


Figure 8-1 I-number relationships

Direct and indirect blocks

In a Berkeley file system, the inode contains 15 disk addresses. The first 12 addresses point to the first 12 logical blocks of the file. These blocks are the **direct data blocks**. If a file contains more than one logical block of data, it continues at the second address to which the inode points. If it contains more than two logical blocks of data, it continues at the third address, and so on, until the first 12 addresses have been used.

If a file has used up the direct data blocks, the 13th address given in the inode is then taken into consideration. The 13th disk address points to an **indirect block**. An indirect block contains the addresses for the next 1024 logical blocks that the file can use. Figure 8-2 illustrates these connections for a 4K file system.

If the file is larger than 1036 blocks (12 blocks for the first 12 addresses, 1024 for the 13th), it continues at the 14th address given in the inode. The 14th address points to a

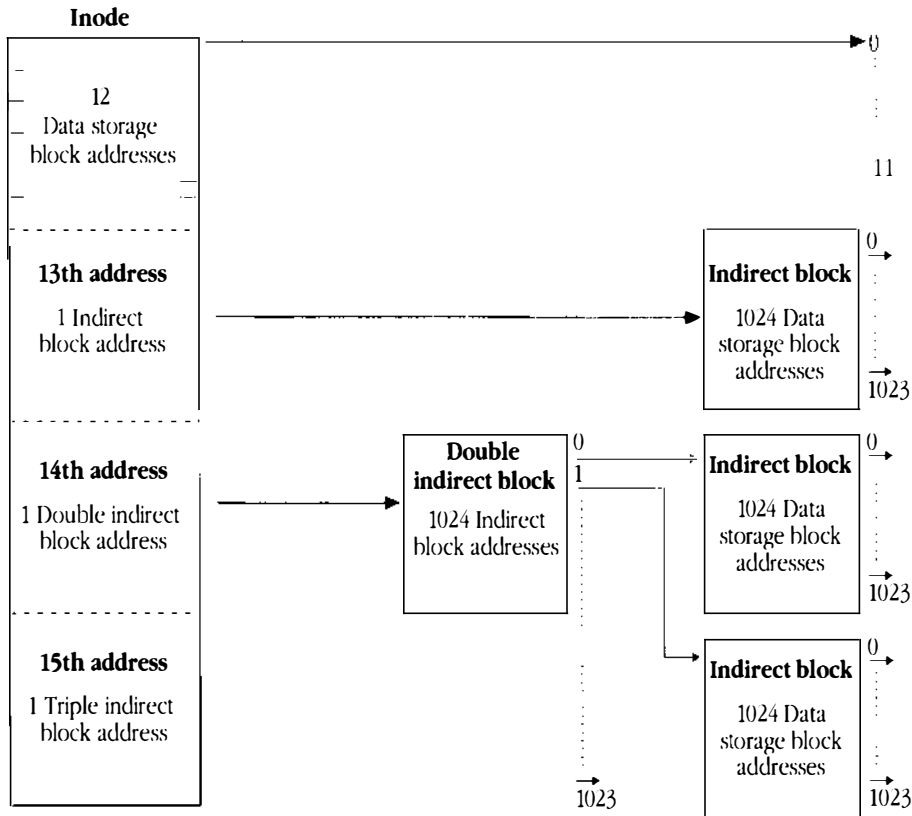


Figure 8-2 Indirect blocks on a 4K file system

double indirect block, which refers to up to 1024 indirect blocks that the file can use. This gives a total of 1,049,612 logical blocks of data: 12 from the first 12 addresses plus 1024 from the indirect block plus 1,048,576 from the double indirect block. For a 1K file system, a file that uses the double indirect block can hold a little over 1 gigabyte of information. On an 8K file system, the same file can theoretically hold about 8.5 gigabytes. However, because an inode's file size is represented in A/UX by a signed 32-bit quantity, a file can never get larger than 2 gigabytes.

The 15th address in the inode points to a **triple indirect block**, which refers to up to 1024 double indirect blocks, each of which in turn refers to up to 1024 indirect blocks, and so on. In practice, the triple indirect block is not used.

System V inodes are similar to the Berkeley model just discussed, though each node only contains 13 disk addresses.

More on inodes

Inodes contain more information than just the location of the data blocks that make up a file. They also contain

- permissions
- owner and group affiliation
- file size
- time the file was last accessed (read)
- time the file was last modified (written)
- time the inode was last modified (written)

Inodes record three different time-related statistics about a file: access time, modification time, and inode modification time. **Access time** is the last time the file was read, and **modification time** is the last time the file was written to.

Inode modification time is sometimes referred to as “creation time.” This is really a misnomer, because modifying, changing permissions, and changing ownership all update the inode modification time on a file.

You can use the `ls -l` command to see some of this additional information. See `ls(1)` for further information on the available options.

Starting from the top

Figure 8-3 illustrates the connection, through multiple inodes, between the root directory and a file called `/usr/tmp/junk` located several levels below the root directory. Initially, the operating system accesses the root directory to get the inode information for the `usr` directory. This gives it the disk location of the `usr` directory, which it then accesses to get the inode information of the `tmp` directory. With this inode information the operating system then locates and accesses the `tmp` directory. This provides it with the inode of the file `junk`, which it then uses to access the first logical block of the file itself.

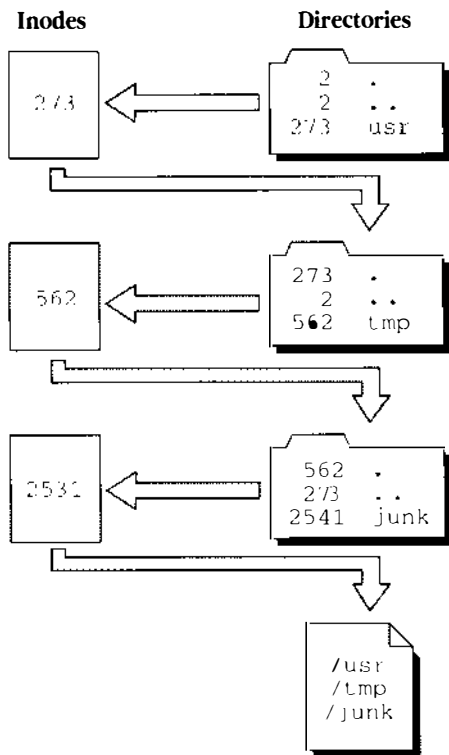


Figure 8-3 File-directory connection through inodes

This process illustrates that a relatively large number of disk accesses are needed to locate a file for the first time. However, because the operating system maintains a cache of active and recently-used inode information in an *inode table*, the next access takes much less time.

Superblock

Each A/UX file system is described by its **superblock**, which is located at the beginning of the file system's disk partition. (UFS file systems maintain copies of this information elsewhere on the disk, as well.) The superblock contains the following critical data about the file system:

- the size of the file system
- the size of the file system's logical blocks
- a magic number that identifies the file system type
- a flag that indicates whether the file system has been mounted read-only
- a flag that indicates whether the superblock has been modified
- a time stamp that shows when the superblock was last modified
- a flag that indicates whether the file system was unmounted cleanly

The UFS file system is subdivided into cylinder groups, each of which contains its own superblock. This superblock contains space for inodes, a copy of the primary superblock in case it is corrupted, and a bitmap indicating whether data blocks are available or in use.

In the SVFS file system, there is only one superblock. It holds the information kept in the UFS cylinder group superblocks. (The inode blocks are actually located on the disk immediately following the System V superblock.) The System V file system uses a free block list to represent available data blocks, instead of the bitmap used in the UFS cylinder group superblocks.

Block I/O

It would be both risky and expensive to keep all data in main memory (also called primary memory, in-core memory, or RAM). Instead, most files are kept in secondary memory (for example, a disk), and the system brings them into main memory as necessary. If you modify files, the system eventually writes the modified versions back into secondary memory for future use.

When a program reads data from or writes data to a disk or tape, the system extracts logical blocks and brings them into main memory. It would be impractical and even unsafe to bring data into main memory without imposing limits and some degree of organization on the amount of data transferred. It would also be highly inefficient for the CPU to do physical input/output (I/O) operations whenever data is transferred from primary to secondary memory and vice versa. For that reason, the system maintains a list of **buffers** for each device. This buffer pool is said to constitute a **buffer cache** for all block-oriented I/O.

The buffer cache

When a program asks the system to read data from a file, the system first searches the cache for the desired block.

If the block is found (for instance, when the system opens a file that is already open), the data is made available to the requesting process without a physical I/O read operation. If the data is not found in the cache, the buffer that has been unused for the greatest period of time is renamed, and data is transferred into it from the disk and made available.

When a process writes a file, the operation occurs in reverse order. A write request first writes data to the buffer cache. Data is written to disk only when the cache is full. Therefore, information about bad write operations refers generally to unusual bad writes to the buffer, and bad disk write operations are generally reported too late to prevent the file system from being corrupted.

Special files and the `/dev` directory

There are several types of files in A/UX: regular files, directories, special files (device files), sockets, symbolic links, and named pipes. In the beginning days of UNIX, only three types of files existed: regular files, directories, and devices. In this context, device files were special and were given the name **special files**. The addition of other file types makes the name no longer appropriate, but it is still used.

When you use the `ls -l` command to list your files, the system response looks like this:

```
-rw-rw---- 1 groupname 13 Sep 25 11:28 file
drwxrwx--- 2 groupname 512 Sep 25 11:28 directory
```

For regular files, such as the first one listed in the example, the first character in the permissions field is `-`. In the case of directories, this character is always `d`. However, suppose that you list `/dev/rdisk/c0d0s0` and `/dev/dsk/c0d0s0` by giving the command

```
ls -l /dev/rdisk/c0d0s0 /dev/dsk/c0d0s0
```

In response, the system displays

```
crw----- 2 bin 24, 0 Mar 25 1992 /dev/rdisk/c0d0s0
brw----- 2 bin 24, 0 Mar 25 1992 /dev/dsk/c0d0s0
```

The first character in the permissions field of `/dev/rdisk/c0d0s0` is `c` (indicating a character device), and the first character in the permissions field of `/dev/dsk/c0d0s0` is `b` (indicating a block device). Either a `b` or a `c` in this position indicates a special (device) file.

Now suppose that you list `/usr/spool/lp/FIFO` by giving the command

```
ls -l /usr/spool/lp/FIFO
```

The `p` in the first field tells you that the file is a pipe:

```
prw----- 1 lp lp 0 Oct 21 1991 /usr/spool/lp/FIFO
```

The other two types of file have their own symbols: `l` (symbolic links) and `s` (sockets).

The contents of device inodes

The files in the `/dev` directory are all special files that the system uses to select a device driver for performing physical I/O.

These files are actually just names and inodes with no associated data on disk (and thus a size of zero bytes). Instead of storing information about the number of bytes in a file, these inodes contain a major and minor number for each special file. These are the numbers you see displayed after you give the `ls -l` command shown in the preceding section.

A device driver is a program that controls the actual physical I/O to the devices listed in the `/dev` directory. However, the device driver itself doesn't reside in the `/dev` directory; rather, it is compiled directly into the kernel.

There is a different device driver for each kind of device (disk drives, tape drives, scanners, and so on). The system uses the **major number** to access the correct device driver. The **minor number** is passed to the driver, which uses this argument to select the correct physical device, including specific attributes of the device. For example, the rewind and no-rewind devices associated with a tape drive device have the same major number (specifying a tape drive) but different minor numbers (specifying the different behaviors).

In summary, the system takes the following steps in response to requests to open special files (such as `fsck` may make):

- looks in `/dev` directory for a file with the requested name
- gets the i-number associated with the filename
- finds the inode specified by the i-number
- gets the major number stored in the inode
- uses this number to select the appropriate device driver
- passes the minor number to the device driver

The driver then uses the minor number to select the correct physical device (the proper partition, in the case of a hard disk device).

Devices (and therefore device drivers and their corresponding special files) come in two forms, block and character (hence the `b` and `c` in the `ls -l` listing). These names refer to the method of I/O used with each type of device.

Block devices such as disks use the block I/O buffer cache mentioned previously and are thus written to, and read from, entire blocks at a time. Character devices such as terminals, line printers, and modems are written to, and read from, one character at a time.

Each hard disk partition is associated with both a character and a block device driver and thus with two special files in the `/dev` directory. For this reason, you can access disk partitions in two ways. They're normally accessed as block devices through the directory hierarchy. But certain programs that access disks, such as `dump.bsd`, `dd`, and `fsck`, run faster when accessing the disk as a character device. For example, `fsck` checks the device `/dev/rdisk/c0d0s0` faster than it checks the device `/dev/dsk/c0d0s0`.

How `fsck` works

As you open, create, and modify files, the system keeps track of all pertinent information about them in two places. This file information—including block sizes, their i-numbers, active and free inodes in the file system, and total number of active, used, and free blocks—is maintained and updated immediately in main memory. Periodically, different parts of this information are written to the disk. If the system crashes, the various file systems can become inconsistent. This inconsistency arises because some file information in main memory is written to disk before the problem while other information is not.

The `fsck` program works by comparing one or more items of information to one or more items of equivalent information. For instance, it compares the number of free blocks available to the number of total blocks in the file system minus the number of blocks in use. If the two numbers are not equal, `fsck` generates an error message. This kind of error results from an inconsistent update or one that was performed out of order. To understand the problems `fsck` is designed to solve, it is helpful to understand these updates.

File system updates

This section describes the various file system updates the system performs every time you create, modify, or remove a file. There are five types of file system updates:

Superblock A mounted file system's superblock is written to disk whenever the file system is unmounted or a `sync(1M)` command is issued. The system periodically issues a `sync(2)` system call to prevent the superblock on disk from getting too out of date.

The superblock of a file system is prone to inconsistency because every change to the blocks or inodes of the file system modifies the superblock.

Inode An inode is written to disk when the file associated with it is closed. In fact, all **in-core blocks** (those pieces of a file in main memory) are also written to disk when a `sync` system call is issued. Thus, the period of danger when inconsistencies can appear is reduced to that between `sync` calls. Typically, the system issues a `sync` call every 30 seconds.

Indirect blocks Indirect blocks, as well as the first twelve blocks of a file, are written to disk whenever they have been modified or released by the operating system. More precisely, they are queued in the buffer cache for eventual writing. Physical I/O is deferred until the A/UX operating system needs the buffer or a `sync` call is issued.

Inconsistencies in an indirect block directly affect the inode that owns it.

Data block A data block is written to disk whenever it has been modified. There are two types of data blocks: plain and directory. **Plain data blocks** contain the information stored in a file. **Directory data blocks** contain directory entries.

The `fsck` program does not attempt to check the validity of the contents of a plain data block. In contrast, `fsck` checks each directory data block for inconsistencies involving the following:

- 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 the dot files (`.` and `..`)
- directories disconnected from the file system

Free list The system updates the free list when a file has been deleted or when a file has been enlarged past a block boundary.

The UFS file system maintains bitmaps that represent the state (allocated/unallocated) of each block contained within the file system. Inconsistencies can still arise if the bitmaps fail to be updated after blocks have been allocated or deallocated from an inode.

For SVFS, the free list begins in the superblock, which contains 49 addresses of blocks available for data storage plus the address of the next free list link block. When the first 49 addresses are used up, the kernel uses the address of the next free list link block to reinitialize the free list in the superblock. As long as disk storage is available, this new list will contain the addresses of the next 49 available free blocks plus the address of the next free list link block. Free list link blocks are chained from the superblock. Therefore, inconsistencies in free list link blocks ultimately affect the superblock.

`fsck` phases

There are six file-system check phases in `fsck` (one of which is generally optional), as well as an initialization phase. Each phase of the `fsck` program passes through the whole file system. If you invoke `fsck` without a device name in the command line, `fsck` repeats all its phases for all devices listed in `/etc/fstab`.

Phase 1: Check blocks and sizes

The `fsck` program checks the inode list. In this phase, `fsck` may discover error conditions that result from checking inode types, setting up the zero-link-count table, checking inode block numbers for bad or duplicate blocks, checking inode size, and checking inode format. Phase 1B runs only if any duplicate blocks (that is, blocks that belong to multiple inodes) are found.

Phase 2: Check pathnames

The `fsck` program removes directory entries pointing to inodes that have error conditions from Phase 1 and Phase 1B. In this phase, `fsck` may discover error conditions that result from root inode mode and status, directory inode pointers out of range, and directory entries pointing to bad inodes.

Phase 3: Check connectivity

The `fsck` program checks the directory connectivity seen in Phase 2. In this phase, `fsck` may discover error conditions that result from unreferenced directories and missing or full `lost+found` directories.

Phase 4: Check reference counts

The `fsck` program reports messages that result from unreferenced files; a missing or full `lost+found` directory; incorrect link counts for files, directories, or special files; unreferenced files and directories; bad and duplicate blocks in files and directories; and incorrect total free inode counts.

Phase 5: Check cylinder groups (UFS only)

This phase is concerned with the free-block and used-inode maps. This section lists error conditions resulting from allocated blocks in the free-block maps, free blocks missing from free-block maps, and the total free-block count incorrect. It also lists error conditions resulting from free inodes in the used-inode maps, allocated inodes missing from used-inode maps, and the total used-inode count incorrect.

Phase 5: Check free list (SVFS only)

The `fsck` program checks each free list link block for a list count out of range, for block numbers out of range, and for blocks already allocated within the file system. A check is made to see that all the blocks in the file system were found.

Phase 6: Salvage free list (SVFS only)

The `fsck` program is concerned with the reconstruction of the free list for SVFS file systems. It lists error conditions resulting from the blocks-to-skip and blocks-per-cylinder values.

Using `fsck`

You can use several options with the `fsck` utility, depending on whether you want to check the root file system or auxiliary file systems.

When to use `fsck`

Any file system inconsistency will be made worse if you continue to use the file system (thus modifying it further) without running `fsck`. Because it is so important to keep your file systems consistent, the `fsck` program is built into the system startup procedure (see Chapter 2, “System Startup and Shutdown,” for a description of the process) and is automatically run each time you start up A/UX.

You are encouraged to run `fsck` any time you suspect the file system integrity, such as after receiving hard read or write errors. Typically, `fsck` is run in single-user state because in this state all file systems (except the root file system) are automatically unmounted. (See Chapter 2 for instructions on how to reach single-user state.) The general syntax of `fsck` is

```
fsck [ options ] [ file-system ]
```

You can specify *options* to direct `fsck` to run in different ways; see `fsck(1M)` in *A/UX System Administrator's Reference* for a complete list of options. You can specify *file-system* to run `fsck` on specific file systems. Without a *file-system* specification, `fsck` runs on the file system names listed in the file `/etc/fstab`; this file is used because it is the list of file systems to be automatically mounted when going to multi-user state.

Note that the file system on which `fsck` is running should be unmounted, or at least no writes should occur while `fsck` is running. This is important because `fsck` performs more than one pass on the file system. If the system is modified from pass to pass, the results are unpredictable.

When `fsck` finds an inconsistency in a file system, it informs you with a message like

```
POSSIBLE FILE SIZE ERROR I 2405
```

The message can also look like this:

```
FREE INODE COUNT WRONG IN SUPERBLK FIX?
```

The second message illustrates one of the interactive error messages produced by `fsck`. The program performs the corrective action only if you enter `y` to confirm that it should do so. If you enter `n`, it will either continue or terminate, depending on the nature of the problem encountered. A later section, “`fsck` Error Messages,” has an exhaustive list of error messages and suggested responses.

`fsck`: a sample interaction

If you bring the system down to single-user state and enter

```
fsck -n
```

`fsck` starts running. Because you didn't specify a file system, `fsck` reads the file `/etc/fstab`. Also, because you invoked `fsck` with the `n` option, `fsck` assumes that you are always answering no to its prompts and thus doesn't open the file system for writing. In other words, the system will not make changes you do not authorize.

◆ **Note** Unless you are familiar with the `fsck` program, always invoke `fsck` the first time with the `-n` option. Read the messages and decide in advance on your course of action before you invoke it a second time without the `-n` option. ◆

For each UFS file system checked, you will see a screen message similar to this one:

```
# fsck -n /dev/dsk/c0d0s0
** /dev/dsk/c0d0s0 (NO WRITE)
** Last Mounted on /
** Root file system
** Phase 1 - Check Blocks and Sizes
** Phase 2 - Check Pathnames
** Phase 3 - Check Connectivity
** Phase 4 - Check Reference Counts
** Phase 5 - Check Cyl groups
4511 file, 39695 used, 11695 free (127 frags, 2892 blocks,
0.2% fragmentation)
```

The `fsck` program lets you know at what phase and in what file system it is at any given time. Different and separate file systems are discussed in the next section, “Multiple File Systems and `fsck`.”

Note that for UFS file systems, the program gives you a measure of the fragmentation of the file system; this tells you how scattered the files are. As a file system becomes increasingly fragmented performance will decrease because the system must spend more time seeking each successive data block. You can defragment the file system by backing up all the files in the file system, removing all the files, then restoring the files from your backups. (The files should not be removed until you have verified that the data on your backups is good, in this instance having two copies of the backups is an excellent idea.) As a general rule, you will see modest performance gains after restoring a file system that is 10 percent fragmented, but significant gains after restoring a file system that is 40 percent fragmented.

The preceding example illustrates a routine check during which no problems or inconsistencies were found. This next example shows an SVFS file system where problems are found. In this situation, you will see a screen message similar to this one:

```

/dev/rdisk/c0d0s1 (NO WRITE)
File System: usr Volume: 003
** Phase 1 - Check blocks and sizes
POSSIBLE FILE SIZE ERROR I=1147
POSSIBLE FILE SIZE ERROR I=1195
** Phase 2 - Check Pathnames
** Phase 3 - Check Connectivity
** Phase 4 - Check Reference Counts
** Phase 5 - Check Free List
1350 files 20582 blocks 18686 free

```

However, the messages can be more obscure and require some action on your part. See “`fsck` Error Messages,” later in this chapter for descriptions of error messages.

Multiple file systems and `fsck`

File-system checks occur when the system goes from single-user to multi-user states. You need to take a few steps to make sure that `fsck` automatically checks file systems other than the root file system during system startup.

Two factors determine whether a file system is checked: options given to the `fsck` command and two fields in the `/etc/fstab` file. As shipped, the system automatically runs `fsck` during startup for the root file system and for files in `fstab`. The determinant fields in the `fstab` file are *type* (of the file system) and *pass-number*. Figure 8-4 shows a sample annotated `/etc/fstab` file. The values it contains causes `fsck` to check the listed file system.

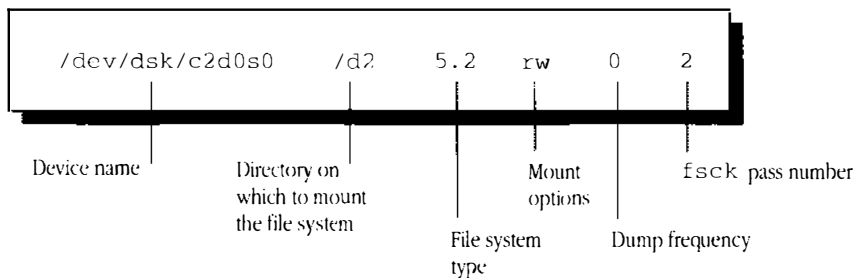


Figure 8-4 A description of sample entries in `/etc/fstab`

Figure 8-5 shows how `fsck` uses its options and the fields of `fstab` to decide whether to check a file system.

You can verify that `fsck` will run automatically at startup on a file system by checking the following:

- the *pass-number* for the file system is a number greater than or equal to 2.
- the *type* of the file system to 4.2 or 5.2. (By default, A/UX makes UFS file systems; their type is 4.2.)

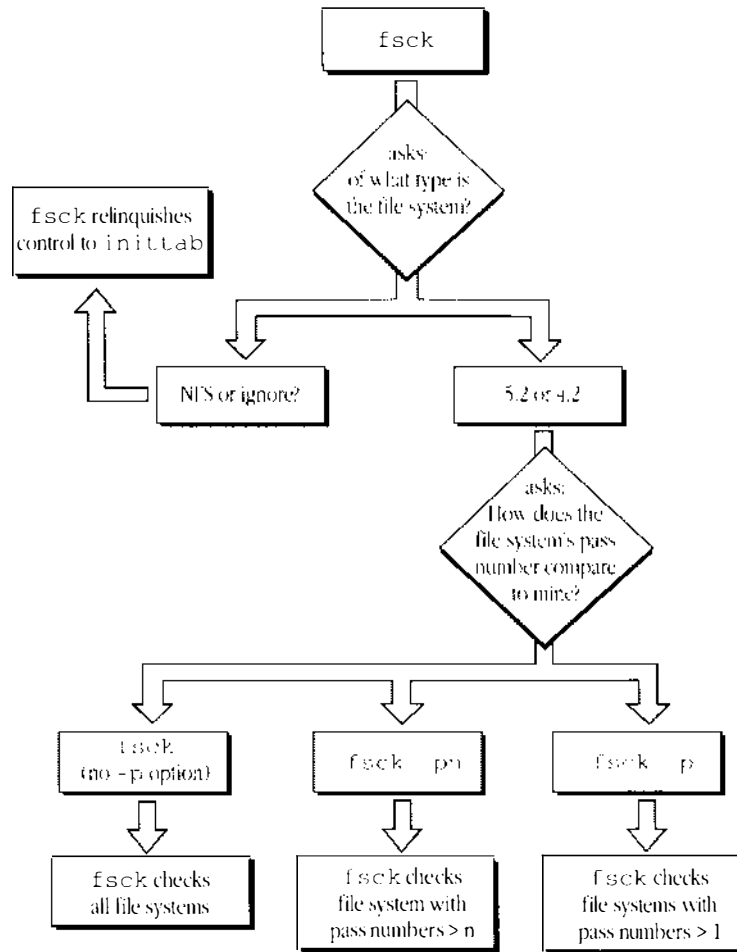


Figure 8-5 How `fsck` decides whether to check a file system

You can have `fsck` check all the file systems having the above attributes (listed in the `/etc/fstab` file) by entering

```
fsck -p
```

Or you can run `fsck` on an individual file system by giving the command `fsck device-name`

`fsck` error messages

The `fsck` program is a multipass file-system check utility. Each file-system pass invokes a different phase of the `fsck` program. After the initial setup, `fsck` performs successive phases 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. However, it is not always possible to correct file system errors by using `fsck` once, as actions taken in later passes can affect the information acquired in earlier passes. Therefore, if errors are encountered you should run `fsck` repeatedly until the file system is error free.

Normally, `fsck` is run noninteractively to correct the file systems after an abrupt halt. These actions are a subset of those that `fsck` takes when it is running interactively. Note that many errors have several responses that you can make.

When an inconsistency is detected, `fsck` reports the error condition to you in a message. If a response is required, `fsck` prints a prompt message and waits for a response. This section explains the possible messages in each phase, the meaning of each message, the possible and suggested responses, and the related error conditions. The suggested responses and the action resulting from the suggested responses are indicated in **bold**.

- ▲ **Warning** Some suggested responses to `fsck` prompts will remove damaged files from your file system. These files may need to be recovered from your backups. It is therefore important to have current backups of all important files. See the section “Restoring Missing Files,” later in this chapter, for information on recovering missing files. ▲

Messages and error conditions are organized by the `fsck` phase in which they can occur. Error conditions that may occur in more than one phase are described in the following section, “`fsck` Initialization Error Messages.”

Some messages require your yes or no response to a prompt. A “yes” response may be `Y` or `y`, and “no” may be `N` or `n`; these responses are shown in uppercase in the following sections. Some messages may indicate a serious problem because the file system cannot be completely checked. If the problem is in software, you may need to remake and restore one or more file systems. If the problem is in hardware, you may need to reinitialize your disk and completely restore all your files from backups.

`fsck` initialization error messages

Before a file system check can be performed, certain files have to be opened. This section discusses error conditions resulting from command line options, memory requests, opening of files, status of files, superblocks, and file-system checks.

`c option?`

In this error message, `c` stands for any character that is not a legal option to `fsck`. Legal options for UFS are `b`, `y`, `n`, and `p`. Legal options for SVFS are `y`, `n`, `s`, `S`, `q`, `D`, and `t`. The `fsck` program terminates on this error condition. Correct the options provided and rerun `fsck`.

`Bad -t option`

This message means that the `t` option was not followed by a filename; `fsck` terminates on this error condition. This message is SVFS-specific.

```
cannot alloc n bytes for blockmap
```

```
cannot alloc n bytes for freemap
```

```
cannot alloc n bytes for statemap
```

```
cannot alloc n bytes for lncntp
```

These messages mean that a request for memory by `fsck` for its virtual memory tables failed. As a consequence, `fsck` terminates. This indicates a serious problem that usually requires restoring the file system being checked. These messages are UFS-specific.

CANNOT READ: BLK *b*

CONTINUE?

This indicates that the `fsck` program's request to read a specified block number *b* in the file system failed.

Possible responses to the `CONTINUE?` prompt are

Y Attempts to continue to run the file system check. The `fsck` program attempts the read again. On UFS file systems, if the read fails, `fsck` displays the message

```
THE FOLLOWING SECTORS COULD NOT BE READ: n
```

where *n* indicates the sectors that could not be read. If `fsck` ever tries to write back one of the blocks on which the read failed, it will print the message

```
WRITING ZERO'ED BLOCK n TO DISK
```

where *n* indicates the sector that was written with zeros. On either kind of file system, if your disk is experiencing hardware problems, the problem will persist. This error condition will not allow a complete check of the file system. Test the disk by using the Hard Disk SC Setup utility, or the formatting utility provided by the manufacturer of your hard drive. Rerun `fsck` to recheck this file system.

N Stops `fsck`.

CANNOT SEEK: BLK *b*

CONTINUE?

This message means that a request to move to a specified block number *b* in the file system failed.

Possible responses to the `CONTINUE?` prompt are

Y Attempts to continue to run the file-system check. If the problem persists, run `fsck` a second time to recheck this file system. Should the problem still remain, test your hard disk drive using the Hard Disk SC Setup utility or the formatting utility provided by the manufacturer of your hard drive.

N Stops `fsck`.

CANNOT WRITE: BLK *b*

CONTINUE?

This indicates that the `fsck` program's request to write a specified block number *b* in the file system failed. The disk is write-protected. This message is UFS-specific.

Possible responses to the `CONTINUE?` prompt are

Y Attempts to continue to run the file-system check. The write operation will be retried, with the failed blocks indicated by the message

```
THE FOLLOWING SECTORS COULD NOT BE WRITTEN: n
```

where *n* indicates the sectors that could not be written. If the disk is experiencing hardware problems, the problem will persist. This error condition will not allow a complete check of the file system. Test the disk by using the Hard Disk SC Setup utility, or the formatting utility provided by the manufacturer of your hard drive. Run `fsck` a second time to recheck this file system.

N Stops `fsck`.

```
CANNOT WRITE: BLK b
```

```
CONTINUE?
```

This message means that an `fsck` program request to write a specified block number *b* in the file system failed. The disk is write-protected. This message is SVFS-specific.

Possible responses to the `CONTINUE?` prompt are

Y Attempts to continue to run the file system check. A second run of `fsck` should be made to recheck this file system. Should the problem remain, test your hard disk drive using the Hard Disk SC Setup utility or the formatting utility provided by the manufacturer of your hard drive.

N Stops `fsck`.

```
can't create f
```

This message means that a request by `fsck` to create a scratch file *f* failed. It ignores this file system and continues checking the next file system given. Check the permissions on the parent directory of *f*. This message is SVFS-specific.

```
can't fstat standard input
```

This message indicates that the attempt to use `fstat` on standard input failed; `fsck` terminates on this error condition. This message is SVFS-specific.

```
can't get memory
```

This means that the `fsck` program can't find the memory space it needs for its virtual memory tables; `fsck` terminates on this error condition. This message is SVFS-specific.

```
can't open f
```

This indicates that the file system *f* cannot be opened for reading. The `fsck` program ignores this file system and continues checking the next file system given. Verify the existence and permissions of *f*, which should be at least 700. If the permissions are correct, try running `fsck` on the file system in single-user mode. Make sure that the drive containing *f* is powered on. If everything else looks good, the file system must be remade and restored.

```
can't open checklist file: f
```

This message means that the default file system check file *f* (usually `/etc/fstab`) cannot be opened for reading; `fsck` terminates on this error condition. Check the permissions of *f*, which should be at least 400, and modify if necessary. If *f* is missing, it can be restored from backups. This message is UFS-specific.

```
can't open: /etc/fstab
```

This error message means that the default file system check file, `/etc/fstab`, cannot be opened for reading; `fsck` terminates on this error condition. If `fstab` exists, check that its permissions include read access; modify if needed. If `fstab` doesn't exist, you can recover it from backups. This message is SVFS-specific.

```
can't stat f
```

```
can't make sense out of name f
```

This indicates that the `fsck` program's request for statistics about the file system *f* failed. It ignores this file system and continues checking the next file system given. Verify the existence and permissions of *f*, which should be at least 700. If the permissions are correct, try running `fsck` on the file system in single-user mode. Make sure that the drive containing *f* is powered on. If everything else looks good, the file system must be remade and restored.

```
can't stat root
```

This message means that the `fsck` program's request for statistics about the root directory (`/`) failed; `fsck` terminates on this error condition. Rerun `fsck` on the root file system in single-user mode. If this doesn't work, the file system must be remade and restored.

IMPOSSIBLE MINFREE=*d* IN SUPERBLOCK
SET TO DEFAULT?

This error indicates that the superblock minimum space percentage is greater than 99 percent or less than 0 percent. This message is UFS-specific.

Possible responses to the SET TO DEFAULT? prompt are

- y** Sets the `minfree` parameter to 10 percent. (If some other percentage is desired, it can be set using `tunefs(8)`.)
- N** Ignores this error condition. One of the following messages appears:

MAGIC NUMBER WRONG
NCG OUT OF RANGE
CPG OUT OF RANGE
NCYL DOES NOT JIVE WITH NCG*CPG
SIZE PREPOSTEROUSLY LARGE
TRASHED VALUES IN SUPER BLOCK

Any of these messages will be followed by the message:

f: BAD SUPER BLOCK: *b*
USE -b OPTION TO FSCK TO SPECIFY LOCATION OF AN ALTERNATE
SUPER-BLOCK TO SUPPLY NEEDED INFORMATION; SEE `fsck(1M)`.

The superblock has been corrupted. An alternative superblock must be selected from among those listed by `newfs(1M)` when the file system was created. For file systems with a blocksize less than 32K (as are the default A/UX file systems), rerun `fsck` using the command: `fsck -b 32`.

Invalid -s argument, defaults assumed

This is only a warning. The `s` option was not followed by 3, 4, or the ratio *blocks-per-cylinder: blocks-to-skip*. The `fsck` program assumes a default value of 400 blocks per cylinder and 9 blocks to skip. This message is SVFS-specific.

Incompatible options: -n and -s

This message reminds you that it is not possible to salvage the free list without modifying the file system; `fsck` terminates on this error condition. This message is SVFS-specific.

f is not a block or character device;OK?

This indicates that the `fsck` program has been given a regular filename by mistake. Verify the type of the file specified or provide the name of a valid file system. This message is UFS-specific.

Possible responses to the OK? prompt are

Y Ignores this error condition.

N Ignores this file system and continues checking the next file system given.

f is not a block or character device

This message means that the `fsck` program has been given a regular filename by mistake; `fsck` ignores this file system and continues checking the next file system given. Rerun `fsck` on the correct file system name. This message is SVFS-specific.

f: (NO WRITE)

This indicates that either the `-n` option was specified or the attempt by `fsck` to open the file system *f* for writing failed. When running manually, all the diagnostics are printed out, but no modifications are attempted to fix them. If the `-n` option was used, this is not an error. If the option was not specified, remove write protection from the appropriate disk. This message is UFS-specific.

Size check: fsize *x* isize *y*

This message means that more blocks are used for the inode list *y* than there are blocks in the file system *x*, or there are more than 65,535 inodes in the file system. The `fsck` program ignores this file system and continues checking the next file system. Remake and restore the file system. This message is SVFS-specific.

UNDEFINED OPTIMIZATION IN SUPERBLOCK
SET TO DEFAULT?

This error message means that the superblock optimization parameter is neither `OPT_TIME` nor `OPT_SPACE`. This message is UFS-specific.

Possible responses to the `SET TO DEFAULT` prompt are

- Y** Sets the superblock to request optimization to minimize running time of the system. (Optimization to minimize disk space utilization can be set using `tunefs(8)`.)
- N** Ignores this error condition.

Phase 1: Check blocks and sizes

Phase 1 is concerned with the inode list. This section lists error conditions resulting from checking inode types, setting up the zero-link-count table, examining inode block numbers for bad or duplicate blocks, checking inode size, and checking inode format.

b BAD I=*i*

This message means that inode *i* contains block number *b* with a number lower than the number of the first data block in the file system or greater than the number of the last block in the file system. This error condition may invoke the `EXCESSIVE BAD BLKS` error condition in Phase 1 (see next message) if inode *i* has too many block numbers outside the file system range. This error condition action always invokes the `BAD/DUP` error condition in Phase 2 and Phase 4.

BAD STATE *ddd* TO BLKERR

This indicates that an internal error has scrambled the `fsck` state map to have the impossible value *ddd*. The `fsck` program exits immediately. This problem can often be solved by rerunning `fsck`. If the problem persists, try autorecovery; if autorecovery doesn't work, the file system being checked must be remade and restored. This message is UFS-specific.

DIRECTORY MISALIGNED T=*i*

This message indicates that the size of directory inode *i* is not a multiple of the size of a directory entry (usually 16). The directory inode should be cleared and the directory restored from backups. This message is SVFS-specific.

b DUP I=*i*

This message means that the inode *i* contains block number *b*, which is already claimed by another inode. This error condition may invoke the `EXCESSIVE DUP BLKS` error condition in Phase 1 if inode *i* has too many block numbers claimed by other inodes. This error condition always invokes Phase 1B and the `BAD/DUP` error condition in Phase 2 and Phase 4.

DUP TABLE OVERFLOW
CONTINUE?

This indicates that an internal table in `fsck` containing duplicate block numbers cannot allocate any more space.

Possible responses to the `CONTINUE?` prompt are

- Y Continues with the program. This error condition will not allow a complete check of the file system. A second run of `fsck` should be made to recheck this file system. If too many duplicate blocks are found, this error condition is repeated.
- N **Stops the program.** Increase the amount of virtual memory available and rerun `fsck`.

EXCESSTVE BAD BLKS I=*i*

CONTINUE?

This message means that the number of blocks 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 associated with inode *i* is too high to be acceptable. Ten is the usual cutoff point.

Possible responses to the CONTINUE? prompt are

- Y** **Ignores the rest of the blocks in this inode and continues checking with the next inode in the file system.** This error condition will not allow a complete check of the file system. Rerun `fsck` to check this file system again.
- N** Stops the program. Rerun `fsck` to check this file system again.

EXCESSTVE DUP BLKS I=*i*

CONTINUE?

This indicates that the number of blocks claimed by other inodes is too high to be acceptable. The cutoff point is usually ten.

Possible responses to the CONTINUE? prompt are

- Y** **Ignores the rest of the blocks in this inode and continues checking with the next inode in the file system.** This error condition will not allow a complete check of the file system. Rerun `fsck` to recheck this file system.
- N** Stops the program. Rerun `fsck` to recheck this file system.

INCORRECT BLOCK COUNT I=*i* (*x* should be *y*)

CORRECT?

This message indicates that the block count for inode *i* is *x* blocks, but should be *y* blocks. This message is UFS-specific.

Possible responses to the CORRECT? prompt are

- Y** **Replaces the block count of inode *i* with *y*.**
- N** Ignores this error condition.

LINK COUNT TABLE OVERFLOW

CONTINUE??

This means that an internal table for `fsck` containing allocated inodes with a link count of zero cannot allocate more memory.

Possible responses to the `CONTINUE??` prompt are

Y Continues with the program. This error condition makes a complete check of the file system impossible. Rerun `fsck` to recheck this file system. If another allocated inode with a zero link count is found, this error condition is repeated.

N Stops `fsck`.

PARTIALLY ALLOCATED INODE: I=*i*

CLEAR??

This error means that inode *i* is neither allocated nor unallocated.

Possible responses to the `CLEAR??` prompt are

Y Deallocates inode *i*. You may need to restore a file from backups.

N Ignores this error condition.

PARTIALLY TRUNCATED INODE: I=*i*

SALVAGE??

This indicates that the `fsck` program has found inode *i* whose size is shorter than the number of blocks allocated to it. This condition should occur only if the system crashes while truncating a file. This message is UFS-specific.

Possible responses to `SALVAGE??` prompt are

Y Completes the truncation to the size specified in the inode.

N Ignores this error condition.

POSSIBLE FILE SIZE ERROR I=*i*

This message indicates that the inode size does not match the actual number of blocks used by the inode. This message is SVFS-specific. If this error occurs, write down the inode number. When `fsck` finishes, continue in single-user mode, mount the file system in which the error occurred, and get its corresponding filename in the following way:

Enter the command

```
ncheck -i i fs
```

where *i* is the number of the inode as provided by the POSSIBLE FILE SIZE ERROR message, and *fs* stands for the name of the file system in which the message occurred.

Important Be sure to use the full pathname of the device, for example `/dev/dsk/c2d0s0`, and not the mount point of the file system.

The `ncheck` command prints the name of the file on the screen. Copy the file to a temporary name and run `sync`. Examine the file and, if you want to retain it, remove the original and give the temporary file its original name. Note that just using `mv` instead of copying the original file will not create a new file and thus will not remove the error condition.

UNKNOWN FILE TYPE I=*i*

CLEAR?

This means that the mode word of the inode *i* indicates that the inode is not a recognized A/UX file type. The problem is usually caused by a corruption of the mode bits.

Possible responses to the CLEAR? prompt are

Y Deallocates inode *i*. This always invokes the UNALLOCATED error condition in Phase 2 for each directory entry pointing to this inode. A file may need to be restored from backups.

N Ignores this error condition.

Phase 1B: Rescan for more duplicates

When a duplicate block is found in the file system, the file system is rescanned to find the inode that previously claimed that block. This section lists the error condition. If the duplicate block is found, the following error condition occurs:

```
b DUP I=i
```

This message means that inode *i* contains block number *b*, which is already claimed by another inode. This error condition always invokes the `BAD/DUP` error condition in Phase 2. You can determine which inodes have overlapping blocks by examining this error condition and the `DUP` error condition in Phase 1.

Phase 2: Check pathnames

This phase is concerned with removing directory entries pointing to inodes that had error conditions from Phase 1 and Phase 1B. This section lists error conditions resulting from root inode mode and status, directory inode pointers in range, directory entries pointing to bad inodes, and directory integrity checks.

```
BAD BLK b IN DIR I=i OWNER=o MODE=m SIZE=s MTIME=t
```

This message occurs only when the `-q` option is used. It means that a bad block was found in inode *i*. This message is SVFS-specific.

This error message indicates that, at a later time, you should either remove the directory inode if the entire block looks bad, or change or remove those directory entries that look bad. You may need to restore several files from backups. The file information provides a clue as to what file may need to be restored.

```
BAD INODE NUMBER FOR '.' I=i OWNER=o MODE=m SIZE=s MTIME=t
DIR=f
FIX?
```

This message means that a directory *i* has been found whose inode number for itself (dot, or `.`) does not equal *i*. This message is UFS-specific.

Possible responses to the `FIX?` prompt are

Y **Changes the inode number for dot (`.`) to be equal to *i*.**

N Leaves the inode number for dot (`.`) unchanged.

```
BAD INODE NUMBER FOR '..' I=i OWNER=o MODE=m SIZE=s MTIME=t
DIR=f
FIX?
```

This indicates that a directory has been found whose inode number for its parent (dot-dot, or `..`) does not equal the parent of *i*. This message is UFS-specific.

Possible responses to the `FIX?` prompt are

Y **Changes the inode number for dot-dot (`..`) to be equal to the parent of dot-dot (`..`).** In the root inode, dot-dot (`..`) points to itself.

N Leaves the inode number for dot-dot (`..`) unchanged.

```
BAD INODE s TO DESCEND
```

This means that an internal error has caused an impossible state *s* to be passed to the routine that descends the file system directory structure. The `fsck` program exits. This problem can often be solved by rerunning `fsck`. If the problem persists, the file system being checked must be remade and restored. This message is UFS-specific.

BAD RETURN STATE *s* FROM DESCEND

This message means that an internal error has caused an impossible state *s* to be returned from the routine that descends the file system directory structure. The `fsck` program exits. This problem can often be solved by rerunning `fsck`. If the problem persists, the file system being checked must be remade and restored. This message is UFS-specific.

BAD STATE *s* FOR ROOT INODE

This error message means that an internal error has caused an impossible state *s* to be assigned to the root inode. The `fsck` program exits. This problem can often be solved by rerunning `fsck`. If the problem persists, the file system being checked must be remade and restored. This message is UFS-specific.

DIRECTORY CORRUPTED I=*i* OWNER=*o* MODE=*m* SIZE=*s* MTIME=*t* DIR=*f*
SALVAGE?

This indicates that a directory with an inconsistent internal state has been found. This message is UFS-specific.

Possible responses to the `FIX?` prompt are

- Y Throws away all entries up to the next directory boundary (usually 512 bytes).** This drastic action can throw away up to 42 entries. You may need to restore several files from backups. The file information provides a clue as to what file or directory may need to be restored.
- N** Skips to the next directory boundary and resumes reading, but does not modify the directory.

DIRECTORY *f* LENGTH *s* NOT MULTIPLE OF *b*
ADJUST?

This means that a directory *f* has been found with size *s* that is not a multiple of the directory blocksize. This message is UFS-specific.

Possible responses to the `ADJUST?` prompt are

- Y The length is rounded up to the appropriate block size.**
- N** Ignores this error condition.


```
DIRECTORY TOO SHORT I=i OWNER=o MODE=m SIZE=s MTIME=t DIR=f
FIX?
```

This indicates that a directory *f* has been found whose size *s* is less than the minimum size directory. The owner *o*, mode *m*, size *s*, modify time *t*, and directory name *f* are printed. This message is UFS-specific.

Possible responses to the `FIX?` prompt are

Y Increases the size of the directory to the minimum directory size.

N Ignores this directory.

```
DUPS/BAD I=i OWNER=o MODE=m SIZE=s MTIME=t FILE=f
REMOVE?
```

This message indicates that during Phase 1 or Phase 1B, `fsck` has found duplicate blocks or bad blocks associated with directory entry *f*, inode *i*. The owner *o*, mode *m*, size *s*, modify time *t*, and filename *f* are printed.

Possible responses to the `REMOVE?` prompt are

Y Removes the directory entry *f*. You may need to restore a file from backups. The file information provides a clue as to what file may need to be restored.

N Ignores this error condition.

```
DUPS/BAD IN ROOT INODE
CONTINUE?
```

This means that during Phase 1 or Phase 1B, `fsck` found duplicate blocks or bad blocks in the root inode (usually inode number 2) for the file system. This message is SVFS-specific.

Possible responses to the `CONTINUE?` prompt are

Y Ignores `DUPS/BAD` error condition in the root inode and attempts to continue to run the file system check. If the root inode is not correct, a large number of other error conditions may result. Rerun `fsck`; if errors still occur remake and restore the file system.

N Stops `fsck`.

DUPS/BAD IN ROOT INODE

REALLOCATE?

This indicates that during Phase 1 or Phase 1B `fsck` has found duplicate blocks or bad blocks in the root inode (usually inode number 2) for the file system. This message is UFS-specific.

Possible responses to the `REALLOCATE?` prompt are

Y Clears the existing contents of the root inode and reallocates it. The files and directories usually found in the root will be recovered in Phase 3 and put into `lost+found`.

If the attempt to allocate the root inode fails, `fsck` exits with the message `CANNOT ALLOCATE ROOT INODE`. If this error message occurs, you have little choice but to restore the file system being checked from backups.

N The `fsck` program prompts with `CONTINUE?`

Possible responses to the `CONTINUE?` prompt are

Y Ignores the `DUPS/BAD` error condition in the root inode and attempts to continue to run the file system check. If the root inode is not correct, then this action may result in many other error conditions.

N Stops the program.

EXTRA '.' ENTRY I=*i* OWNER=*o* MODE=*m* SIZE=*s* MTIME=*t* DIR=*f*
FIX?

This error message means that a directory *i* has been found that has more than one entry for dot (.). This message is UFS-specific.

Possible responses to the `FIX?` prompt are

Y Removes the extra entry for dot (.).

N Leaves the directory unchanged.

```
EXTRA '..' ENTRY I=i OWNER=o MODE=m SIZE=s MTIME=t DIR=f  
FIX?
```

This means a directory *i* has been found that has more than one entry for dot-dot (..). This message is UFS-specific.

Possible responses to the `FIX?` prompt are

Y Removes the extra entry for dot-dot (..).

N Leaves the directory unchanged.

```
n IS AN EXTRANEOUS HARD LINK TO A DIRECTORY d  
REMOVE?
```

This message indicates that the `fsck` program has found a hard link, *n*, to a directory, *d*. This message is UFS-specific.

Possible responses to the `REMOVE?` prompt are

Y Deletes the extraneous entry, *n*.

N Ignores the error condition.

```
MISSING '..' I=i OWNER=o MODE=m SIZE=s MTIME=t DIR=f  
FIX?
```

This means that a directory *i* has been found whose first entry is unallocated. This message is UFS-specific.

Possible responses to the `FIX?` prompt are

Y Builds an entry for dot (.) with inode number equal to *i*.

N Leaves the directory unchanged.

```
MISSING '..' I=i OWNER=o MODE=m SIZE=s MTIME=t DIR=f  
CANNOT FIX, FIRST ENTRY IN DIRECTORY CONTAINS f
```

This indicates that a directory *i* has been found whose first entry is *f*. The `fsck` program cannot resolve this problem. Mount the file system and move the entry *f* elsewhere. Unmount the file system and run `fsck` again. This message is UFS-specific.

MISSING '.' I=*i* OWNER=*o* MODE=*m* SIZE=*s* MTIME=*t* DIR=*f*

CANNOT FIX, INSUFFICIENT SPACE TO ADD '.'

This message means that a directory inode *i* (with name *f*) has been found whose first entry is not dot (.). The `fsck` program cannot resolve this problem. The directory inode must be cleared and the directory restored from backups. This message is UFS-specific.

MISSING '..' I=*i* OWNER=*o* MODE=*m* SIZE=*s* MTIME=*t* DIR=*f*

FIX?

This message indicates that a directory *i* has been found whose first entry is unallocated. This message is UFS-specific.

Possible responses to the `FIX?` prompt are

Y Builds an entry for dot-dot (..) with inode number equal to the parent of dot-dot (..). In the root inode, dot-dot (..) points to itself.

N Leaves the directory unchanged.

MISSING '..' I=*i* OWNER=*o* MODE=*m* SIZE=*s* MTIME=*t* DIR=*f*

CANNOT FIX, SECOND ENTRY IN DIRECTORY CONTAINS *f*

This indicates that a directory *i* has been found whose second entry is *f*. The `fsck` program cannot resolve this problem. Mount the file system and move the entry *f* elsewhere. Unmount the file system and run `fsck` again. This message is UFS-specific.

MISSING '..' I=*i* OWNER=*o* MODE=*m* SIZE=*s* MTIME=*t* DIR=*f*

CANNOT FIX, INSUFFICIENT SPACE TO ADD '..'

This means that a directory inode *i* (with name *f*) has been found whose second entry is not dot-dot (..). The `fsck` program cannot resolve this problem. The directory inode must be cleared and the directory restored from backups. This message is UFS-specific.

NAME TOO LONG *f*

This message means that an excessively long pathname has been found. This usually indicates loops in the file system name space. This can occur if a user logged in as root has made circular links to directories. The file system must be reinitialized and restored from backups. The circular links can be removed by a technical expert, though the procedure is beyond the scope of this book. This message is UFS-specific.

i OUT OF RANGE I=*i* NAME=*f*

REMOVE?

This indicates that a directory entry *f* has an inode number that is greater than the end of the inode list.

Possible responses to the REMOVE? prompt are

Y Removes the directory entry *f*. You may need to restore a file from backups.

N Ignores this error condition.

ROOT INODE NOT DIRECTORY

REALLOCATE?

This message means that the root inode (usually inode number 2) is not of the directory type. This message is UFS-specific.

Possible responses to the REALLOCATE? prompt are

Y Clears the existing contents of the root inode and reallocates it. The files and directories usually found in the root inode are recovered in Phase 3 and put into the directory `lost+found`.

If the attempt to allocate the root inode fails, `fsck` exits with the message `CANNOT ALLOCATE ROOT INODE`. If this error message occurs, you have little choice but to restore the file system being checked from backups.

N The `fsck` program prompts with FIX?

Possible responses to the FIX? prompt are

Y Makes the root inode's type a directory. If the root inode's data blocks are not directory blocks, many error conditions are produced.

N Stops the program.

```
ROOT INODE NOT DIRECTORY
```

```
FIX?
```

This indicates that the root inode (usually inode number 2) is not of the directory type. This message is SVFS-specific.

Possible responses to the `FIX?` prompt are

Y **Makes the root inode type a directory.** If the root inode's data blocks are not directory blocks, a very large number of error conditions result.

N Stops `fsck`.

```
ROOT INODE UNALLOCATED
```

```
ALLOCATE?
```

This message means that the root inode (usually inode number 2) has no allocate mode bits. This message is UFS-specific.

Possible responses to the `ALLOCATE?` prompt are

Y **Allocates inode 2 as the root inode.** The files and directories usually found in the root are recovered in Phase 3 and put into `lost+found`. If the attempt to allocate the root fails, `fsck` exits with the message `CANNOT ALLOCATE ROOT INODE`.

If this error message occurs, you have little choice but to remake and restore the file system being checked from backups.

N Causes `fsck` to exit.

```
ROOT INODE UNALLOCATED. TERMINATING
```

This means that the root inode (always inode number 2) has no allocated mode bits. This error condition indicates a serious problem that you may not be able to solve without remaking and restoring the file system being checked. The program ends.

```
UNALLOCATED I=i OWNER=o MODE=m SIZE=s MTIME=t DIR=f
```

```
REMOVE?
```

This message indicates that a directory or file entry *f* points to an unallocated inode *i*. The owner *o*, mode *m*, size *s*, modify time *t*, and directory entry *f* are printed. This message is UFS-specific.

Possible responses to the `REMOVE?` prompt are

- Y Removes the directory entry *f*.** You may need to restore a file or directory from backups. The file information provides a clue as to what file may need to be restored.
- N** Ignores this error condition.

```
UNALLOCATED T=i OWNER=o MODE=m SIZE=s MTIME=t NAME=f
REMOVE?
```

This indicates that a directory entry *f* has an inode *i* without allocated mode bits. The owner *o*, mode *m*, size *s*, modify time *t*, and filename *f* are printed. If the file system is not mounted and the `-n` option wasn't specified, the entry is removed automatically if the inode it points to contains size 0. This message is SVFS-specific.

Possible responses to the `REMOVE?` prompt are

- Y Removes the directory entry *f*.** You may need to restore a file from backups. The file information provides a clue as to what file may need to be restored.
- N** Ignores this error condition.

```
ZERO LENGTH DIRECTORY I=i OWNER=o MODE=m SIZE=s MTIME=t DIR=f
REMOVE?
```

This message means that a directory entry *f* has a size *s* that is zero. The owner *o*, mode *m*, size *s*, modify time *t*, and directory name *f* are printed. This message is UFS-specific.

Possible responses to the `REMOVE?` prompt are

- Y Removes the directory entry *f*.** This action always invokes the `BAD/DUP` error condition in Phase 4. You may need to restore a directory from backups. The file information provides a clue as to what directory may need to be restored.
- N** Ignores this error condition.

Phase 3: Check connectivity

Phase 3 is concerned with the directory connectivity seen in Phase 2. This section lists error conditions that result from unreferenced directories and missing or full `lost+found` directories.

`BAD INODE s TO DESCEND`

This message indicates that an internal error has caused an impossible state *s* to be passed to the routine that descends the file system directory structure. The `fscck` program exits. This problem can often be solved by rerunning `fscck`. If the problem persists, the file system being checked must be remade and restored. This message is UFS-specific.

`DIR I=i CONNECTED. PARENT WAS I=j`

This advisory message indicates that a directory inode *i* was successfully connected to the `lost+found` directory. The parent inode *j* of the directory inode *i* is replaced by the inode number of the `lost+found` directory. You can use the command

```
find / -i j -print
```

to determine the former parent directory's name, then move the directory with inode *i* back to its normal location.

`DIRECTORY f LENGTH s NOT MULTIPLE OF b
ADJUST?`

This means that a directory *f* has been found with size *s* that is not a multiple of the directory blocksize *b* (this can reoccur in Phase 3 if it is not adjusted in Phase 2). This message is UFS-specific.

Possible responses to the `ADJUST?` prompt are

- Y** **The length is rounded up to the appropriate block size.** A warning is printed and the directory is adjusted.
- N** Ignores the error condition.

lost+found IS NOT A DIRECTORY
REALLOCATE?

This indicates that the entry for `lost+found` is not a directory. This message is UFS-specific.

Possible responses to the `REALLOCATE?` prompt are

Y Allocates a directory inode and changes `lost+found` to refer to it. The previous inode reference by the `lost+found` name 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 generates the message

```
SORRY. CANNOT CREATE lost+found DIRECTORY
```

and stops the attempt to link up the lost inode. This action always invokes the `UNREF` error condition in Phase 4.

N Stops the attempt to link up the lost inode. This action always invokes the `UNREF` error condition in Phase 4.

NO lost+found DIRECTORY
CREATE?

This means that there is no `lost+found` directory in the root directory of the file system; `fsck` can try to create one. This message is UFS-specific.

Possible responses to the `CREATE?` prompt are

Y Creates a `lost+found` directory in the root directory of the file system. This may raise the message

```
NO SPACE LEFT IN / (EXPAND).
```

See the message later in this section for the possible responses. Inability to create a `lost+found` directory generates the message

```
SORRY. CANNOT CREATE lost+found DIRECTORY
```

and stops the attempt to link up the lost inode. This action always invokes the `UNREF` error condition in Phase 4.

N Stops the attempt to link up the lost inode. This action always invokes the `UNREF` error condition in Phase 4.

```
NO SPACE LEFT IN /lost+found DIRECTORY
EXPAND?
```

This means there is no space to add another entry to the `lost+found` directory in the root directory of the file system. This message is UFS-specific.

Possible responses to the `EXPAND?` prompt are

Y **Expands the `lost+found` directory to make room for the new entry.** If the attempted expansion fails, `fsck` prints the message

```
SORRY. NO SPACE IN lost+found DIRECTORY
```

and stops the attempt to link up the lost inode. This action always invokes the `UNREF` error condition in Phase 4. Clean out unnecessary entries in `lost+found`.

N Stop the attempt to link up the lost inode. This action always invokes the `UNREF` error condition in Phase 4.

```
SORRY. NO lost+found DIRECTORY
```

This indicates that there is no `lost+found` directory in the root directory of the file system. In this case, `fsck` ignores the request to link a directory in `lost+found`.

This action always invokes the `UNREF` error condition (described later). If

`lost+found` exists, check its permissions. See `fsck(1M)` and

`mkllost+found(1M)` in *A/UX System Administrator's Reference* for further details. This message is SVFS-specific.

```
SORRY. NO SPACE IN lost+found DIRECTORY
```

This message means that there is no space in the `lost+found` directory to add another entry; `fsck` ignores the request to link a directory in `lost+found`. This action always invokes the `UNREF` error condition (described later). Clean out unnecessary entries in the `lost+found` directory, or make it larger. See `fsck(1M)` in *A/UX System Administrator's Reference* for further details. This message is SVFS-specific.

```
UNREF DIR I=i OWNER=o MODE=m SIZE=s MTIME=t
RECONNECT?
```

This message indicates that the directory inode *i* was not connected to a directory entry when the file system was traversed. The owner *o*, mode *m*, size *s*, and modify time *t* of the directory inode are printed. While being checked, the directory is reconnected if its size is nonzero; otherwise it is cleared.

Possible responses to the `RECONNECT?` prompt are

- Y Reconnects the directory inode *i* to the file system in the directory for lost files (usually `lost+found`).** This may invoke the `lost+found` error condition in Phase 3 if there are problems connecting the directory inode to `lost+found`. This may also invoke the `CONNECTED` error condition in Phase 3 if the link was successful.
- N** Ignores this error condition. This action always invokes the `UNREF` error condition in Phase 4.

Phase 4: Check reference counts

Phase 4 is concerned with the link count information of Phase 2 and Phase 3. This section lists error conditions resulting from unreferenced files; missing or full `lost+found` directory; incorrect link counts for files; unreferenced files and directories; and bad or duplicate blocks in files and directories. All errors in this phase except for running out of space in the `lost+found` directory are correctable.

```
BAD/DUP FILE I=i OWNER=o MODE=m SIZE=s MTIME=t
CLEAR?
```

This message indicates that during Phase 1 or Phase 1B, `fsck` has found duplicate blocks or bad blocks associated with file inode *i*. The owner *o*, mode *m*, size *s*, and modify time *t* of inode *i* are printed.

Possible responses to the `CLEAR?` prompt are

- Y Deallocates inode *i*.** You may need to restore the file from backups. The file information provides a clue as to what file may need to be restored.
- N** Ignores this error condition.

```
BAD/DUP DIR I=i OWNER=o MODE=m SIZE=s MTIME=t
CLEAR?
```

This means that during Phase 1 or Phase 1B, `fsck` found duplicate blocks or bad blocks associated with directory inode *i*. The owner *o*, mode *m*, size *s*, and modify time *t* of inode *i* are printed. This message is SVFS-specific.

Possible responses to the `CLEAR?` prompt are

- Y Deallocates inode *i*.** You may need to restore a file from backups. The file information provides a clue as to what file may need to be restored.
- N** Ignores this error condition.

```
FREE INODE COUNT WRONG IN SUPERBLK
FIX?
```

This indicates that the actual count of the free inodes doesn't match the count in the superblock of the file system. If you specify the `-q` option, the count is fixed automatically in the superblock. This message is SVFS-specific.

Possible responses to the `FIX?` prompt are

- Y Replaces the count in superblock with the actual count.**
- N** Ignores this error condition.

```
LINK COUNT FILE I=i OWNER=o MODE=m SIZE=s MTIME=t COUNT=x
SHOULD BE y
ADJUST?
```

This means that the link count for inode *i*, which is a file, is *x* but should be *y*. The owner *o*, mode *m*, size *s*, and modify time *t* are printed. The `fsck` program exits with the message `LINK COUNT INCREASING`. This message is UFS-specific.

Possible responses to the `ADJUST?` prompt are

- Y Replaces the link count of file inode *i* with *y*.**
- N** Ignores this error condition.

```
LINK COUNT DIR I=i OWNER=o MODE=m SIZE=s MTIME=t COUNT=x
SHOULD BE y
ADJUST?
```

This message indicates that the link count for inode *i*, which is a directory, is *x* but should be *y*. The owner *o*, mode *m*, size *s*, and modify time *t* of directory inode *i* are printed. This message is SVFS-specific.

Possible responses to the ADJUST? prompt are

Y Replaces the link count of directory inode *i* with *y*.

N Ignores this error condition.

```
LINK COUNT f I=i OWNER=o MODE=m SIZE=s TIME=t COUNT=x
SHOULD BE y
ADJUST?
```

This message means that the link count for file *f* inode *i* is *x* but should be *y*. The filename *f*, owner *o*, mode *m*, size *s*, and modify time *t* are printed. This message is SVFS-specific.

Possible responses to the ADJUST? prompt are

Y Replaces the link count of inode *i* with *y*.

N Ignores this error condition.

```
lost+found IS NOT A DIRECTORY
REALLOCATE?
```

This means that the entry for `lost+found` is not a directory. This message is UFS-specific.

Possible responses to the REALLOCATE? prompt are

Y **Allocates a directory inode and changes `lost+found` to refer to it.** The previous inode reference by the `lost+found` name is not cleared. It is either reclaimed as an unreferenced inode or has its link count adjusted later in this phase. Inability to create a `lost+found` directory generates the message

```
SORRY. CANNOT CREATE lost+found DIRECTORY
```

and stops the attempt to link up the lost inode. This action always invokes the UNREF error condition in Phase 4. One or more files may need to be restored from backups.

N Stops the attempt to link up the lost inode. This action always invokes the UNREF error condition in Phase 4. One or more files may need to be restored from backups.

NO lost+found DIRECTORY
CREATE?

This indicates that there is no `lost+found` directory in the root directory of the file system; `fsck` ignores the request to link a file in `lost+found`. See the entry for `mklost+found(1M)` in *A/UX System Administrator's Reference* for additional information. This message is UFS-specific.

Possible responses to the `CREATE?` prompt are

- Y** **Creates a `lost+found` directory in the root of the file system.** This may raise the message: `NO SPACE LEFT IN / (EXPAND)`.

See below for the possible responses. Inability to create a `lost+found` directory generates the message

```
SORRY. CANNOT CREATE lost+found DIRECTORY
```

and stops the attempt to link up the lost inode. This action always invokes the `UNREF` error condition in Phase 4. One or more files may need to be restored from backups.

- N** Stops the attempt to link up the lost inode. This action always invokes the `UNREF` error condition in Phase 4. One or more files may need to be restored from backups.

NO SPACE LEFT IN lost+found DIRECTORY
EXPAND?

This means that there is no space to add another entry to the `lost+found` directory in the root directory of the file system; `fsck` ignores the request to link the file. Check the size and contents of `lost+found`. This message is UFS-specific.

Possible responses to the `EXPAND?` prompt are

- Y** **Expands the `lost+found` directory to make room for the new entry.** If the attempted expansion fails, `fsck` prints the message

```
SORRY. NO SPACE IN lost+found DIRECTORY
```

and stops the attempt to link up the lost inode. This action always invokes the `UNREF` error condition in Phase 4. Clean out unnecessary entries in `lost+found`. One or more files may have to be restored from backups.

- N** Stops the attempt to link up the lost inode. This action always invokes the `UNREF` error condition in Phase 4. One or more files may have to be restored from backups.

SORRY. NO `lost+found` DIRECTORY

This indicates that there is no `lost+found` directory in the root directory of the file system; `fsck` ignores the request to link a file in `lost+found`. This action always invokes the `CLEAR?` error condition (described later). Check the permissions of `lost+found`. Also see `mklost+found(1M)` in *A/UX System Administrator's Reference*. This message is SVFS-specific.

SORRY. NO SPACE IN `lost+found` DIRECTORY

This message means that there is no space to add another entry to the `lost+found` directory in the root directory of the file system; `fsck` ignores the request to link the file. This action always invokes the `CLEAR?` error condition (described next). Check the size and contents of `lost+found`. This message is SVFS-specific.

CLEAR?

The inode mentioned in the preceding error condition cannot be reconnected.

Possible responses to the `CLEAR?` prompt are

- y** **Deallocates the inode mentioned in the preceding error condition.** You may need to restore a file from backups.
- N** Ignores this error condition.

UNREF DIR I=*i* OWNER=*o* MODE=*m* SIZE=*s* MTIME=*t*

CLEAR?

This message indicates that the inode *i*, which is a directory, was not connected to a directory entry when the file system was traversed. The owner *o*, mode *m*, size *s*, and modify time *t* of inode *i* are printed. If you don't set the `-n` option and the file system is not mounted, empty directories are cleared automatically. Nonempty directories are not cleared. This message is SVFS-specific.

Possible responses to the `CLEAR?` prompt are

- y** **Deallocates inode *i*.** You may need to restore a directory from backups. The file information provides a clue as to what file may need to be restored.
- N** Ignores this error condition.

UNREF FILE I=*i* OWNER=*o* MODE=*m* SIZE=*s* MTIME=*t*

RECONNECT?

This means that inode *i* was not connected to a directory entry when the file system was traversed. The owner *o*, mode *m*, size *s*, and modify time *t* of inode *i* are printed. If the `-n` option is not set and the file system is not mounted, empty files are not reconnected and are cleared automatically.

Possible responses to the `RECONNECT?` prompt are

- Y Reconnects inode *i* to the file system in the directory for lost files, usually `lost+found`.** This may invoke a `lost+found` error condition (described in the following section) if there are problems connecting inode *i* to `lost+found`.
- N** Ignores this error condition. This action always invokes the `CLEAR?` error condition (described next).

CLEAR?

The inode mentioned in the immediately preceding error condition cannot be reconnected.

Possible responses to the `CLEAR?` prompt are

- Y Deallocates the inode mentioned in the immediately preceding error condition.** A file may need to be restored from backups.
- N** Ignores this error condition.

UNREF FILE I=*i* OWNER=*o* MODE=*m* SIZE=*s* MTIME=*t*

CLEAR?

This indicates that inode *i*, which is a file, was not connected to a directory entry when the file system was traversed. The owner *o*, mode *m*, size *s*, and modify time *t* of inode *i* are printed. If you don't set the `-n` option and the file system is not mounted, empty files are cleared automatically.

Possible responses to `CLEAR?` prompt are

- Y Deallocates inode *i*.** You may need to restore a file from backups. The file information provides a clue as to what file may need to be restored.
- N** Ignores this error condition.

Phase 5: Check cylinder groups (UFS only)

This phase is concerned with the free-block and used-inode maps. This section lists error conditions resulting from allocated blocks in the free-block maps, free blocks missing from free-block maps, and the total free-block count incorrect. It also lists error conditions resulting from free inodes in used-inode maps, allocated inodes missing from used-inode maps, and incorrect total used-inode counts.

```
BLK(S) MISSING IN BIT MAPS  
SALVAGE?
```

This message means that a cylinder group block map is missing some free blocks; the maps are reconstructed. This message is UFS-specific.

Possible responses to the `SALVAGE?` prompt are

Y Reconstructs the free block map.

N Ignores this error condition.

```
CG c: BAD MAGIC NUMBER
```

This indicates that the magic number of cylinder group `c` is wrong. This usually indicates that the cylinder group maps have been destroyed. When you run the `fsck` command at the A/UX command line, the cylinder group is marked as needing to be reconstructed. This message is UFS-specific.

```
FREE BLK COUNT(S) WRONG IN SUPERBLOCK  
SALVAGE?
```

This message indicates that the superblock free block information is incorrect and asks if it should be recomputed. This message is UFS-specific.

Possible responses to the `SALVAGE?` prompt are

Y Reconstructs the superblock free block information.

N Ignores this error condition.

SUMMARY INFORMATION BAD

SALVAGE?

This means that the summary information was found to be incorrect. This message is UFS-specific.

Possible responses to the `SALVAGE?` prompt are

Y **Reconstructs the summary information.**

N Ignores this error condition.

Phase 5: Check free list (SVFS only)

The free list starts in the superblock and continues through the free list link blocks of the file system. Each free list link block can be checked for a list count out of range, for block numbers out of range, and for blocks already allocated within the file system. A check is made to be sure that all the blocks in the file system were found.

The free list check begins in the superblock. The `fsck` program checks the list count for a value of less than 0 or greater than 50. It also checks each block number for a value of less than the first data block in the file system or greater than the last block in the file system. Then it compares each block number to a list of already allocated blocks. If the free list link block pointer is nonzero, `fsck` reads in the next free list link block and repeats the process.

When all the blocks have been accounted for, `fsck` checks whether the number of blocks used by the free list plus the number of blocks claimed by the inodes equals the total number of blocks in the file system. If anything is wrong with the free list, `fsck` may rebuild the list, excluding all blocks in the list of allocated blocks.

The superblock also contains a count of the total number of free blocks in the file system. The `fsck` program compares this count to the number of blocks it found free in the file system. If the counts do not agree, `fsck` may replace the count in the superblock with the actual free block count.

During Phase 5, `fsck` lists error conditions resulting from bad blocks in the free list, bad free-block count, duplicate blocks in the free list, unused blocks from the file system not in the free list, and incorrect total free-block count.

`x BAD BLKS IN FREE LIST`

This means that the free list contains `x` blocks with a block number less than the first data block in the file system or greater than the last block in the file system. This error condition always invokes the `BAD FREE LIST` condition.

`BAD FREEBLK COUNT`

This indicates that the count of free blocks in a free list link block is greater than 50 or less than 0. This error condition always invokes the `BAD FREE LIST` condition.

`BAD FREE LIST`

`SALVAGE?`

This message indicates that during Phase 5, `fsck` has found bad blocks in the free list, duplicate blocks in the free list, or blocks missing from the file system. If the `-q` option is specified, the free list is salvaged automatically.

Possible responses to the `SALVAGE?` prompt are

Y **Replace the actual free list with a new free list.** The new free list is ordered to shorten the time spent waiting for the disk to rotate into position.

N Ignores this error condition.

`x BLK(S) MISSING`

This message means that the free list does not contain `x` blocks unused by the file system. This error condition always invokes the `BAD FREE LIST` condition.

`x DUP BLKS IN FREE LIST`

This indicates that the free list contains `x` blocks claimed by inodes or earlier parts of the free list. This error condition always invokes the `BAD FREE LIST` condition.

EXCESSIVE BAD BLKS IN FREE LIST
CONTINUE?

This means that the free list contains more than a tolerable number of blocks with a value less than the first data block in the file system or greater than the last block in the file system. Usually ten is the cutoff point.

Possible responses to the CONTINUE? prompt are

Y **Ignores the rest of the free list and continues execution of fsck.** This error condition always invokes the BAD BLKS IN FREE LIST error condition.

N Stops fsck.

EXCESSIVE DUP BLKS IN FREE LIST
CONTINUE?

This message means that the free list contains more than a tolerable number of blocks claimed by inodes or earlier parts of the free list. Usually ten is the cutoff point.

Possible responses to the CONTINUE? prompt are

Y **Ignores the rest of the free list and continues execution of fsck.** This error condition always invokes the DUP BLKS IN FREE LIST error condition.

N Stops fsck.

FREE BLK COUNT WRONG IN SUPERBLK
FIX?

This indicates that the actual count of free blocks does not match the count in the superblock of the file system.

Possible responses to the FIX? prompt are

Y **Replaces the count in the superblock with the actual count.**

N Ignores this error condition.

Phase 6: Salvage free list (SVFS only)

This phase is concerned with the free list reconstruction. During this phase, `fsck` lists error conditions resulting from the blocks-to-skip and blocks-per-cylinder values.

Default free list spacing assumed

This advisory message indicates that the blocks-to-skip value is greater than the blocks-per-cylinder value, the blocks-to-skip value is less than 1, the blocks-per-cylinder value is less than 1, or the blocks-per-cylinder value is greater than 500. The default values of 9 blocks-to-skip and 400 blocks-per-cylinder are used. See `fsck(1M)` in *A/UX System Administrator's Reference* for further details.

Cleanup

Once a file system has been checked, a few cleanup functions are performed. The `fsck` program lists advisory messages about the file system and its modify status.

`v files, w used, x free (y frags, z blocks,
f % fragmentation)`

This advisory message indicates that the file system checked contained v files using w fragment-sized blocks, leaving x fragment-sized blocks free in the file system. The numbers in parentheses break the free count down into y free fragments and z free full-sized blocks. This message is UFS-specific.

`x files y blocks z free`

This advisory message indicates that the file system checked contained x files using y blocks, leaving z blocks free in the file system. This message is SVFS-specific.

***** REBOOT A/UX! *****

This advisory message indicates that the root file system has been modified by `fsck`. A/UX restarts the system. This message is UFS-specific.

```
***** Fixed root filesystem, rebooting A/UX! *****
```

This message indicates that `fsck` has modified the root file system to fix inconsistencies it found. The `fsck` program forces a restart because it can fix the file system image only on the disk but not in memory. This message is SVFS-specific.

```
***** FILE SYSTEM WAS MODIFIED *****
```

This advisory message indicates that the current file system was modified by `fsck`. If this file system is mounted or is the current root file system, `fsck` should be halted and A/UX restarted.

Restoring missing files

Occasionally, files on your system may become corrupted and must be removed, either manually or by a system utility such as `fsck`. You will not always know what files have been removed; sometimes the only information you have about a file being removed is its inode number. If the file is a critical system file, it may be replaced by autorecovery. More likely, however, the corrupted file will be a user file and you will have to restore the missing file from your backups. In general, you must take the following steps to restore files that are missing:

- Determine the set of files that remain on your system.
- Determine the set of files backed up on your media.
- Calculate the difference between the two sets.
- Use the difference information to restore the missing files.

Fortunately, two of the utilities provided with A/UX, `pax` and `cpio`, can perform all these functions for you. And, since `pax` can read both `cpio` and `tar` archives, it can be used with backups made with any of the `cpio`, `tar`, or `pax` utilities.

The `restore` utility, while it does not automatically restore missing files, does allow you to extract files by inode number. This can be helpful when the inode number is the only information you have about a removed file.

There are also several Macintosh backup utilities that can automatically restore missing files. However, unless a backup utility has been certified to run with A/UX, file permission and ownership information may be lost.

Restoring with `cpio`

The `cpio` utility will not overwrite a newer file with an older version unless directed to do so. Therefore, you can simply restore `cpio` backups and missing files are the only ones that will be extracted. This can be handy if one of the non-critical system files is missing, and you need to restore the file from your A/UX 3.0 Installation CD-ROM.

For example, to restore (but not overwrite) any missing files in the `/bin` directory from the A/UX 3.0 CD-ROM (in a drive at SCSI ID 3), enter the commands:

```
mount /dev/dsk/c3d0s0 /mnt.  
cd /mnt/bin  
find . -depth -print | cpio -pdlm /bin
```

◆ **Note** Be careful when restoring directories from the A/UX 3.0 Installation CD-ROM. It is easy to restore much more data than you expect, especially packages that you may not have installed previously. ◆

You can also use the `cpio` command to restore any missing files from your backups. See Chapter 5 of this guide, “Backing Up Your System,” and `cpio(1)` for further information.

Restoring with `restore`

The `restore` utility allows you to extract files from a backup by the file's inode number. You may want to use this option if the `fsck` program has removed corrupt files but only tells you the inode of the files removed.

A typical command line for restoring inodes from a floppy disk is

```
restore -xmf /dev/rfloppy0 5468 3299 423
```

See Chapter 5 of this guide and `restore(1M)` for further information.

Restoring with `pax`

The `pax` command has an update option (`-u`) that will not overwrite a newer file with an older version. Therefore, you can simply restore `pax` backups using this option and the missing files are the only ones that will be extracted. The `pax` utility can also read `tar` archives directly and `cpio` archives created with the `-c` option.

A typical command line for restoring (updating) files from a floppy disk into the current directory is:

```
pax -r -o -u -f /dev/fd/d0ew
```


A typical command line for restoring (updating) files from a cartridge tape into the current directory is:

```
tcb < /dev/rmt/tc1 | pax -r -o -u
```

See Chapter 5 of this guide and `pax(1)` for further information.



9 System Activity Package






The system activity counters / 9-3

The system activity data collector / 9-3

Setting up the system activity functions / 9-5

The system activity reports / 9-6





The system activity package, located in the `/usr/adm/sa` directory, provides features for collecting data about the low-level functioning of your system. You can use commands to get information on low-level system activity and to generate reports that summarize this activity. The system activity package does this by using various counters to monitor kernel activity. The counters are sampled regularly, and the data is saved in binary format. This data is then used to generate ASCII reports, which can help you determine if and where the system needs fine-tuning. You can view the output from these commands immediately or redirect it to a file for future use.

The system activity commands use a series of activity counters to gather and sample data and generate a report on system activity. These counters are described in conjunction with the commands that use them to generate reports.

When originally implemented for large, multiterminal UNIX installations, the system activity package provided information that was used to “tune” the kernel, optimizing its parameters for each particular installation. However, A/UX dynamically allocates many of these adjustable parameters, removing the need for kernel tuning. Therefore, the primary use of the A/UX system activity package is for benchmarking and development.

The system activity counters

A series of system activity counters must be working to determine what processes are running at any given time. These counters, which are located in the operating system kernel, record various activities at selected times. For example, you can set them to record all processes used at 8:00 A.M. on Monday, and store the information in a file for later review.

There are several types of counters. Each counter generates various pieces of information, but the same counter may be used by different commands to provide different information. The **CPU counter**, for instance, reports the prime-time and nonprime time usage in minutes in the accounting report (see Appendix D, “System Accounting Package”), whereas in the activity report the same counter reports the state or states that the CPU is in: idle, user, kernel, or wait.

In this chapter, each type of counter is explained as it first occurs in relation to the system command that invokes it. Most are explained in relation to the `sar` command, because this is the most useful system activity command.

If you would like more detailed information about the counters, keep the following in mind:

- The data structure for most counters is defined in the `sysinfo` structure in `/usr/include/sys/sysinfo.h`.
- The system table overflow counters are kept in the `_syserr` structure.
- The device activity counters come from the device status tables.

The system activity data collector

The system activity data collector (`sadc`) is the automated data collection feature supplied with your system. Two shell scripts, `sa1` and `sa2`, assemble the data for the reporting functions.

The `sadc` command

The `sadc` command collects system activity information at regular intervals. It is an executable program that reads the system counters located in `/dev/kmem` and records them in binary format in a file for later sampling by the system activity reporting functions.

The command `sadc` works alone or with arguments. It has the format

```
/usr/lib/sa/sadc [t n] [file]
```

When used without arguments, `sadc` sets the startup time. It creates a special record that tells the system to reset the system counters to zero. The same thing occurs when the system is restarted. When the arguments `t` and `n` are used, `sadc` samples the counters `n` times, with `t` seconds between each sample, and writes the data in binary format to the named *file* or, by default, to the standard output, `/usr/adm/sa/sadd`, where *dd* stands for the current day.

The `sa1` and `sa2` commands

The `sa1` and `sa2` commands supply the default parameters for the operation of `sadc` and `sar`, respectively.

The `sa1` shell script invokes `sadc` to enter the information gathered by the system counters at the intervals specified in the `/usr/lib/cron/crontab/adm` file into the daily data file `/usr/adm/sa/sadd`. This information is in binary format.

The `sa2` shell script invokes the `sar` command. It reads the data file created by `sa1` and uses it to generate a report in ASCII format. This report is stored in the `/usr/adm/sa` directory in the files named `sardd`. Again, *dd* stands for the day of the report. This script also removes those report or data files in the `/usr/adm/sa` directory that are more than seven days old.

You can use the `sar` options with the `sa2` command. For a complete explanation of these options, see “The System Activity Reporter,” later in this chapter.

Setting up the system activity functions

If you want to track system activity, it is a good idea to monitor and record the activity routinely in a standard way for historical analysis. By automating these functions, you can generate regular system activity reports. The `crontab` function is used to automate these tasks; see the section “Automating System Administration with `crontab`,” in Chapter 5, for additional information.

Your system can automatically sample the activity counters and store the information in a file for later reporting. Here is the procedure for activating the system activity functions:

1 Log in as root.

2 Change your effective user name to *adm*.

Enter the command:

```
su adm
```

3 Copy the current crontab file.

Enter the command:

```
crontab -l > /tmp/adm.cron
```

to put a copy of the current crontab file into the file `/tmp/adm.cron`.

4 Edit the file `/tmp/adm.cron` to reflect the new changes. Remove the leading # from the following lines:

```
0 * * * 0,6 /usr/lib/sa/sa1
0 18-7 * * 1-5 /usr/lib/sa/sa1
0 8-17 * * 1-5 /usr/lib/sa/sa1 1200 3
0 20 * * 1-5 /usr/lib/sa/sa2 -s 8:00 -e 18:00 -i 3600 -uyba
```

The first three entries produce activity records at midnight on weekends, every 20 minutes (three samples, 1200 seconds apart) between the hours of 8:00 A.M. and 5:00 P.M. during weekdays, and on the hour at other times. The last entry compiles a single report at 8:00 P.M. of each working day giving an hourly summary of all activity in the interval 8:00 A.M. to 6:00 P.M.

5 **Notify the operating system to read the modified file.**

Enter the command:

```
crontab /tmp/adm.cron
```

This copies the file into `/usr/spool/cron/crontabs/adm` and alerts `cron` to reread the crontab file. The edited entries cause the collection functions to operate. It is the responsibility of the system administrator to check the desired reporting function regularly and to copy reports for later review.

The system activity reports

The commands `sar` and `timex` generate system activity reports. An additional command, `sag`, can create graphs of system activity on specialized terminals.

You can use these commands to observe system activity during

- normal operations
- a controlled stand-alone benchmark test of a program
- an uncontrolled run of a program to observe the operating environment

The `sar` command generates system activity reports in real time and saves the output in a file.

The `timex` command is a modified `time` command that reports how long a given command takes to execute and how much user and system time was spent in execution of the command.

The `sag` command displays system activity in graph form, though this is not supported on Macintosh computers.

These commands and their options and applications are discussed in detail in the following sections.

System activity reports

The various reports you can get from `sar` and the options used to invoke them are listed in the following subsections.

The system activity reporter

The system activity reporter command is `sar`. You can use the `sar` command alone or with a series of options. If you enter

```
sar
```

a report on today's CPU activity scrolls across the screen. (See the section "CPU Utilization," later in this chapter, for an example of the report.) By entering `sar` with the options discussed in the next sections, you can sample various counters to view activity at specific times and intervals. By redirecting the output to a file, you can save the information for later review.

The full `sar` command syntax is written in one of these two ways:

```
sar [-option]... [-stime] [-etime] [-isec] [-f file]
```

```
sar [-option]... [-ofile] t [n]
```

The first word is the command itself; you select one or more *options* that sample different system counters. The remaining options allow you to specify sampling times and save the results for later viewing. The available *options* are listed in `sar(1M)`; reports that the *options* produce are described later in this section.

In the command syntax

```
sar [-option]... [-stime] [-etime] [-isec] [-f file]
```

`sar` extracts the data from a previously recorded file (which you specify with the `-f file` option) or, by default, from the standard system activity daily data file `/usr/adm/sa/sadd`, where `dd` stands for the current day. This file appears unreadable when you view it because it is stored in the internal format used by the system reporting functions to prepare reports. The reports themselves are in a readable format.

You can specify the starting and ending times of the report by invoking the `-s time` and `-e time` options respectively, using the format `hh[:mm[:ss]]`.

The `-i` option selects records at *sec* second intervals. Otherwise, all intervals found in the data file are reported. For example,

```
sar -s8:00 -e18:01 -i3600
```

samples the stored data at intervals of 3600 seconds, or every hour starting at 8:00 A.M. and ending at 6:01 P.M. You cannot, of course, get more frequent samples than have been stored in the data file.

In the command syntax

```
sar [-option]... [-ofile] t [n]
```

`sar` invokes the data collection program `sadc` to sample the system activity counters *n* times, at intervals of *t* seconds, and generate a system activity report. The `-o` option redirects the report to *file*.

If you supply no frequency argument (*t*), `sar` generates a system activity report at the time interval recorded in the existing data file. See “Setting Up the System Activity Functions,” earlier in this chapter, for a description of the sampling frequency. Unless you redirect the output to a file, it scrolls across the screen.

◆ **Note** All reports you generate print a time stamp for each entry. This time stamp appears in the first column of each report section in the format *hh:mm:ss*. ◆

When used without *options*, the `sar` command generates a report about CPU activity only. This is the same report you receive if you enter

```
sar -u
```

You can get a report containing all activity information by typing

```
sar -A
```

This generates the same information as

```
sar -udqbwcaym
```

but takes fewer keystrokes.

The system activity package, using the `sa2` shell script, automatically generates a report containing all of the options described in the following sections. These reports are stored in the `/usr/adm/sa` directory, in the files listed as `sardd`.

CPU utilization

Use the `-u` option to get a report on CPU utilization. This is the default option, the option the system selects if none is specified.

If you enter

```
sar -u
```

a report similar to the following is displayed:

07:00:02	%usr	%sys	%wio	%idle
08:00:04	0	1	0	99
09:00:02	10	7	3	80
11:00:07	8	8	4	81
12:00:05	15	10	4	71
13:00:03	12	10	2	76
15:00:08	31	16	4	48
16:00:06	39	15	4	43
18:00:02	1	1	0	97
19:00:02	0	1	0	99
Average	12	8	2	78

The column headings are defined as follows:

%usr	percent of time running in user mode
%sys	percent of time running in system mode
%wio	percent of time idle with process waiting for block I/O
%idle	percent of time idle with process not waiting for block I/O

All of the information under these headings is sampled each hour to produce the report. The headings correspond to the four CPU counters: user, kernel, wait for I/O completion, and idle. These counters are increased by one (incremented) each time the clock calls for an interrupt, which occurs 60 times per second.

Buffer activity

The `-b` option reports buffer activity. This option works by accessing three sets of read and write counters:

- `lread` and `lwrite` (logical read, logical write)
- `bread` and `bwrite` (block read, block write)
- `phread` and `phwrite` (physical read, physical write)

If you enter

```
sar -b
```

the system generates a report organized in columns as follows:

```
bread/s, bwrit/s
```

Samples the `bread` and `bwrite` counters, giving the transfers of data per second between the system buffers and the disk.

```
lread/s, lwrit/s
```

Samples the `lread` and `lwrite` counters, giving the number of times the system buffers are accessed.

```
phread/s, phwrit/s
```

Samples the `phread` and `phwrite` counters, giving the number of data transfers via raw devices.

```
%rcache, %wcache
```

Gives the **cache hit ratio**, the ratio of buffer reads (`bread`) to logical reads (`lread`) and buffer writes (`bwrit`) to logical writes (`lwrit`). The cache hit ratio reports whether files are being accessed from the buffers or information is being retrieved from the disk itself. A low ratio might suggest that increasing the number of buffers could increase system response time.

Swapping and switching activity

The `-w` option reports swapping and switching activity. This option uses the `swpin` and `swpout` counters. (The column headings that appear in the report, `swpin` and `swpot`, are abbreviations.) These counters are incremented each time the system receives a request initiating a transfer to or from the swap device. The **swap device** is a disk partition used as a “holding area” for processes that are not currently running. When main memory is full, processes that are not currently running are swapped out (transferred) to the swap device. When the CPU is ready to work on a process that is in the swap device, the process is swapped back into memory.

The amount of data swapped in and out is measured in blocks and counted by the `bswapin` and `bswapout` counters. The data collected by these counters is displayed under the column headings `bswin` and `bswot`.

The figures for `swpin` can be higher than those for `swpot`. This peculiar asymmetry arises from programs with the sticky bit set, which keeps a program in main memory.

If you enter `sar -w`, the system generates a report organized in columns as follows:

<code>swpin/s</code>	Reports the number of transfers from the swap area on disk to main memory.
<code>swpot/s</code>	Reports the number of transfers from memory to the swap area.
<code>bswin/s, bswot/s</code>	Reports the number of bytes transferred.
<code>pswch/spswch/s</code>	Reports the number of process switches that have occurred.

System call activity

The `-c` option reports system calls. This option accesses the `pswitch` and `syscall` counters. These counters are related to the management of multiprogramming, which occurs when one process, the **parent process**, calls (forks and executes) another program, the **child process**. While the parent waits, the child performs its task, terminates, and reinvokes the parent process, which continues.

The `syscall` counter is incremented every time a system call occurs. Certain system calls—the `read`, `write`, `fork`, and `exec` calls—are counted individually in `sysread`, `syswrite`, `sysfork`, and `sysexec`.

The `pswitch` counter keeps track of the number of times the switcher is invoked. The switcher is invoked when the program running cannot complete its intended process.

The following situations cause `pswitch` to be incremented:

- a system call that had to wait for an unavailable resource
- an interrupt that caused the awakening of a higher-priority process
- a 1-second clock interrupt

To check the number of system calls, type

```
sar -c
```

The first five columns of the resulting output give information on the number of system calls made. The first column (`scall/s`) gives the total number of system calls; the next four columns give the number of specific system calls for read (`sread/s`), write (`swrite/s`), fork (`fork/s`), and execute (`exec/s`) system calls. The last two columns give the number of characters transferred by the read (`rchar/s`) and write (`wchar/s`) system calls.

Queue activity

The `-q` option reports on queue activity. At intervals of one second, the clock routine examines the process table to see whether any processes are queued and ready. If so, the counter `runocc` is incremented and the number of processes waiting is added to the `runque` counter. While this is happening, the clock routine also checks the process status of the swapper. If the swap queue is occupied, the counter `swapocc` (swap occupied) is incremented and the number of processes waiting in the queue is added to the `swapque` counter.

If you enter

```
sar -q
```

the system reports on the average time a process is queued before it is acted on. The report lists the average queue length while the queue is occupied (the columns ending in `-sz`) and the percentage of time the queue is occupied (the columns beginning with `%`). It is broken down into two main parts: the run queue (the `runq` columns), which lists the processes in memory and runnable; and the swap queue (the `swp` columns), which lists the processes swapped out but ready to run.

Message and semaphore activity

The `-m` option reports message and semaphore activities. This option of the `sar` command reports which processes requested the operating system to send information directly to another process.

If you enter

```
sar -m
```

the system generates a report on message and semaphore activity. The message and semaphore columns (`msg/s` and `sema/s`) reflect the most basic I/O operations of the system. These “primitives” (for example, `read` and `write`) are called by other programs, which use them as building blocks to complete their processes. The message primitives keep track of interprocess communications; that is, the number of times one process asks the operating system to send information directly to another process. The semaphore primitives synchronize the actions of various processes and facilitate the use of shared resources.

Device activity

The `-d` option reports device activity. Because the system activity is sampled infrequently as compared with the actual device activity, this option is unreliable and the information it provides is usually inaccurate. When you select this option, you can view information on block devices, such as the disk drives.

If you enter

```
sar -d
```

the system generates a report organized in columns as follows:

Device	The device being sampled.
%busy	Samples the device counters and displays the time, expressed as a percentage of total report time, during which the device was engaged in transferring data.
avque	Displays the average number of requests waiting to be processed.
r+w/s	Displays the number of data transfers to or from the disk drive.
blks/s	Displays the number of bytes transferred and counted in block-sized units.
await	Displays the number of milliseconds that transfer requests have to wait in the queue before being processed.
avserv	Displays the average time it takes to service the request.

The figures in these columns represent a sampling of a combination of I/O activity counters and character counters. The %busy column, for example, represents a sampling of the io_ops counters and the io_act counters. The blks/s column represents a sampling of the io_bcnt counters. The avserv and await columns represent a sampling of the io_act and io_resp counters. These counters are explained in greater detail later in this section, and some suggestions for fine-tuning are given.

Each disk or tape device has four counters to record activity. The activity information is kept in the device status table. Whenever an I/O request occurs, io_ops (I/O operations) is incremented. It keeps track of block I/O, swap I/O, and physical I/O.

Transfers between the device (particular disk or tape drive) and memory are recorded in 512-byte blocks by io_bcnt (I/O block counters). The io_act and io_resp are particularly useful I/O counters. They measure the time it takes a device to receive, process, and transmit a request, summed over all I/O requests for the device. (Time is measured in ticks; each tick is 1/60 of a second.) The io_act counter measures the active time, which is the time during which the device is actively engaged in seeking, rotating, and transferring data (all measured by different counters and combined into one active count).

The `io_resp` counter measures the total elapsed time between the time the request is received in the queue and the time it is completed. By looking at the ratio between active time and response time, you can determine if the disk devices are being put to their best use. For example, if one disk is so heavily used that response time is significantly increased, perhaps you can shorten system reaction time by transferring some frequently used information to another, less used disk. Also, if the active time counter for a disk is high compared with the number of requests on the system, perhaps you can load the file systems on the device differently for more effective access.

Table overflow status

The `-v` option reports the status of text, process, inode, and file tables. Because of the sampling frequency, the information provided is usually inaccurate. When an overflow occurs in any of the inode, file, text, or process tables, the corresponding counter (`inodeovf`, `fileovf`, `textovf`, or `procovf`) is incremented. These indicate resource problems with tables or with the size of memory.

You can use the `-v` option of the `sar` command to discover the size of tables and any table overflows.

If you enter

```
sar -v
```

the system produces a report in which the first five columns show the number of used and available entries in each table. This information is typically given for intervals of one hour. The measurement is taken once at the sampling point. The last four columns give the number of overflows that occur between the sampling points.

File access activity

The `-a` option reports on use of file access system routines. This information is usually inaccurate because this option reports only on System V file systems, and the default file systems used with A/UX are UFS. This option of `sar` checks the following:

- the number of times a file's inode number is requested
- the number of file path searches
- the number of directory blocks read by the system

If you enter

```
sar -a
```

the system samples the file access counters and generates a report showing the number of times each of the file access routines was performed. The report is organized into columns as follows:

<code>iget/s</code>	Measures the number of requests for the inode number that corresponds to a particular file. The <code>iget</code> routine is used to locate the inode entry (i-number) of a file. (See Chapter 8, “Checking the A/UX File System: <code>fsck</code> ,” for an explanation of i-numbers and inodes.) The <code>iget</code> routine first searches the in-core (main memory) inode table. If the inode entry is not in the table, <code>iget</code> gets the inode from the file system where the file resides and makes an entry in the inode table.
<code>namei/s</code>	Measures the number of requests for a file-system path search. The <code>namei</code> routine performs file-system path searches. It searches the various directory files to get the associated i-number of a file corresponding to a special path. Like other file access routines, <code>namei</code> calls <code>iget</code> to find the i-number of the file it is searching for. Therefore, counter <code>iget</code> is always greater than counter <code>namei</code> .
<code>dirblk/s</code>	Measures and records the number of directory block read requests issued by the system. Dividing the directory blocks read by the number of <code>namei</code> calls results in an estimate of the average path length of files. A long path length may indicate a significant number of subdirectories. Rearranging the file structure to move the more commonly accessed files higher up the path would correct this problem.

Each time one of these routines is called, the respective counter is incremented.

Single-command activity report

The `timex` command is an extension of the `time` command; see `time(1)` and `timex(1)` in *A/UX Command Reference*. It times a command and reports process data and system activity. The command you are tracking is executed, and the elapsed time, user time, and system time spent in execution are reported in seconds.

The options available with `sar` are also available with `timex`. However, `timex` tracks one command, whereas `sar` tracks all commands. The output of the `timex` command is easier to understand than the output of `sar`, and it is also generally more reliable.

Normally you use the `timex` command to measure a single command. If you want to measure multiple commands, combine the commands in an executable file and time the file. You can also do this by entering

```
timex sh -c "cmd1; cmd2; ...;"
```

This allows `timex` to measure the user and system times consumed by all the commands as if they were one single command. See `sh(1)` in *A/UX Command Reference*.

Because process records associated with a command are selected from the accounting file `/usr/adm/pacct`, background processes that have the same user ID, terminal ID, and execution time window are included in the totals given.

You can specify options to list or summarize process accounting data for the command and its children and to report the total system activity during the execution interval. If you don't specify any options, `timex` behaves exactly as the `time` command does.

The syntax for the `timex` command is

```
timex [-pos] command
```

The `timex` command options are

- p Lists the process accounting records for the specified command.
This option has six suboptions:
 - f Prints the `fork/exec` flag and system exit status.
 - h Reports the fraction of total available CPU time the process consumes during its execution and suppresses reporting of the mean memory size.
 - k Reports the total kcore-minutes and suppresses reporting of memory size.
 - m Reports the mean core size.
 - r Reports the fractional representation of CPU factor (user time over system time plus user time).
 - t Reports separate system and user CPU times.
- o Reports the total number of blocks read or written and total characters transferred by the command selected and all its child processes.
- s Reports the total system activity during the execution interval of the command, not just the activity resulting from the command specified. All the data items listed in `sar` are reported.

System activity graphs

Graphs of system activity can be created by editing the ASCII data produced by various system activity reports and importing the result into a third-party Macintosh charting application. Alternately, you can write an `awk` filter that will do the editing for you.

The UNIX system activity graph command is `sag`, which displays the system activity data created by a previous run of the `sar` command and stored in binary format. You can plot the graph using any single column or combination of columns since `sag` can prepare cross-plots or time plots.

A graphics package that can invoke the `graphics` and `tplot` commands must reside on the system for you to print a system activity graph. Unfortunately, very few terminals are supported. The Macintosh II and common emulators such as the VT100 are not among those supported. See `sag(1G)` in *A/UX Command Reference*.

10 Troubleshooting

Trouble while starting up A/UX / 10-3

Accessing the A/UX file system from A/UX Startup / 10-4


Reinitializing a disk with bad sectors / 10-5

Screen locks, power failures, and emergencies / 10-6

Troubleshooting user and group administration / 10-7

Errors while changing your password / 10-10

Other miscellaneous issues / 10-11



This chapter lists some common problems that A/UX users may experience, along with actions that you can take to identify and correct them. Error messages about line printer systems, `fsck`, and autorecovery are available in the chapters that discuss those subjects.

Problems that involve A/UX not starting up or requiring single-user mode will usually be resolved through the use of the A/UX Startup utilities. Other problems that are contained in a particular subset of A/UX, such as a command not behaving as you expect, will usually be fixed from the running A/UX system.

Trouble while starting up A/UX

If you are having trouble while you are starting up the system, it may help you to understand how A/UX starts up from a hard disk.

How A/UX starts up from a hard disk

A/UX is designed to be started up from any SCSI hard disk, from SCSI ID 0 through SCSI ID 6. The A/UX kernel accepts startup device information from A/UX Startup. The A/UX kernel requires two pieces of information about the disk from which it will run: the root file system and the swap area. It calls the disk block device drivers (by using the major number to select one of the block device drivers), and passes to that driver a parameter (the minor number) indicating the partition within a particular disk in which to look.

When the A/UX Startup application is launched, it finds the A/UX kernel and loads it into main memory. It leaves in main memory a set of major and minor numbers for A/UX to use for the root file system and swap area.

Troubleshooting start up problems

You may view messages displayed during the startup process by clicking the Messages button when it is displayed in the startup progress screens. You may also change the AutoLaunch command in the Booting dialog box of A/UX Startup to use the `launch -v` command. Both these actions will provide you with more information about what is happening on the system, and may help you pinpoint the problem area. If the information displayed does not help you figure out what may be wrong, be sure to copy down all the information as it appears on the screen. You will need these messages when describing the problem to others.

Accessing the A/UX file system from A/UX Startup

This is a very useful procedure to fix a multitude of problems, from creating a new `/etc/passwd` file to fixing entries in the `/etc/inittab` file. Remember that there is a limited set of commands available within A/UX Startup; for instance, you may not use wildcards and all editing must be done with the `ed` editor.

1 Find the A/UX Startup command line interface.

Double-click the A/UX Startup icon and press `COMMAND-PERIOD`. Enter a password, if needed, to get an A/UX Startup window.

2 Find the SCSI ID number of the disk you want to access.

Most likely this will be the disk containing the root file system of A/UX. To find out where the root file system resides, look at the Root Directory box in the General dialog box in A/UX Startup. For more information, see Chapter 2, “System Startup and Shutdown.”

3 Use the `chroot` command to get access to that SCSI device.

If you have not changed the configuration of A/UX from the standard release, you will be in the root directory of A/UX when you enter A/UX Startup. If you want to access the root directory in another disk, use the command

```
chroot (ID, 0, 0)
```

where *ID* is the SCSI ID of the disk you want to access.

4 Use the `fsck` command on the file systems you may alter.

Never edit files on a file system without assuring the consistency of that file system by first running the `fsck` command.

5 **Edit the file(s) you wish; save the changes.**

Since there is no full screen editor available in A/UX Startup, you need to use `ed`—a line editor. If you have accidentally destroyed an important file such as `/etc/passwd` or `/etc/inittab`, you only need to rebuild enough of the file to get you to single-user mode, where you can replace the file with a backup. It is even possible to use the `echo` command to do this, by redirecting the output of `echo` into a file.

6 **Restart or continue the A/UX start up process.**

This action depends on what you have changed in A/UX Startup. If you are not sure, go ahead and restart the system. If you use the command `launch -v`, the system gives more information about what is happening during the boot process, and prompts you for any required information the system may require.

Reinitializing a disk with bad sectors

A situation in which you may need to create an A/UX file system is when you need to correct disk errors created by bad sectors on a disk. Hard Disk SC Setup can be used to map out bad sectors by reinitializing the disk. (If Hard Disk SC Setup does not work on your drive, use the partitioning utility provided by the manufacturer.) The following is a list of overall steps you take to reinitialize an error-prone disk.

1 **Run the file system consistency check, `fsck(1M)`.**

Normally, `fsck` can fix most data storage inconsistencies on the disk, but you may have to run it few times before it no longer reports errors. Procedures are provided in Chapter 8, “Checking the A/UX File System: `fsck`.” If errors remain after several attempts to run `fsck`, proceed to step 2.

2 **Back up your entire system.**

The utilities `cpio` and `dump.bsd` have the ability to resynchronize themselves after encountering a hard disk error, and thus are preferable to use in this situation. Procedures are provided in Chapter 5, “Backing Up Your System.”

3 **Test the disk using Hard Disk SC Setup.**

This tests the drive for media errors. If you are still experiencing disk errors, you may have no choice but to reinitialize the disk. Proceed to step 4.

4 **Reinitialize and partition the disk.**

Follow the procedures given in Chapter 4, “Adding and Managing Hard Disk SCs,” in *Setting Up Accounts and Peripherals for A/UX*.

5 **Remake each of the file systems on the disk.**

6 **Restore the original contents of each of the file systems using the most recent backups available.**

Screen locks, power failures, and emergencies

If your system freezes and you cannot enter commands, you can try the following techniques to fix the problem:

- If you have modified `/etc/inittab` to start a `getty` on another terminal, you can try to log in as root from another terminal.
- If you are connected to a network, attempt to log in over it.
- If you have MacsBug installed, and one of your Macintosh applications freezes, you may be able to use MacsBug troubleshooting techniques to stop the program. MacsBug is a debugging program that can be used with A/UX. Two useful commands you can enter are
 - `es`, which terminates the currently running (or halted) program
 - `rb`, which terminates the Finder
- Pressing **COMMAND-CONTROL-E** terminates all active Macintosh applications, if possible.
- Sometimes simply waiting a few minutes unlocks the screen.

As a last resort, press the power switch on the rear of the computer or reset the system by using the Macintosh programmer's switch.

- ▲ **Warning** Turning off the system power or using the programmer's switch should be your last resort because this can cause problems with the A/UX file system, and can either damage or lose data stored on the hard disk. ▲

In a normal shutdown process (sometimes called a *graceful shutdown*), any data in the disk write buffer is written to the disk and all disk file information is saved before the file systems are unmounted and the power is turned off. In an emergency or if the power goes off, an ungraceful shutdown is your only choice.

File system inconsistencies are detected and repaired, when possible, by the file system check command, `fsck`. This program runs automatically when the system boots, and if it detects problems it displays a message and asks whether to repair the problem:

```
The file system at /mnt needs repairs. Repair automatically?  
The best action is always to choose automatic repair. If you wish to use fsck manually  
you can run the fsck command while in the A/UX Startup program and see exactly  
where the problems are occurring. Chapter 8, "Checking the A/UX File System: fsck,"  
provides additional information about fsck.
```

Troubleshooting user and group administration

Most user administration problems can be traced to ownership and group membership questions or to erroneous entries in the `/etc/passwd` and `/etc/group` files.

Suggestions for solving these potential problems, indicated by alert boxes and messages, follow. The message that the system displays is given first, followed by a description which tells you what action to take.

Sorry, that user name is unknown. Please retype the name or contact the system administrator.

The Name field in the Login dialog box has a name that isn't listed in `/etc/passwd`.

Sorry, your password is incorrect. Please reenter it.

The password for that user is different than what is recorded in the `/etc/passwd` file.

Your home directory, [*name of home directory*], is inaccessible. Perhaps that directory is on a file system which is not mounted. Please contact the system administrator.

The user's home directory (as listed in `/etc/passwd`) can't be found. Another possibility is that the system administrator made a directory in which the name differed from that in the `/etc/passwd` file.

Your default shell program, [*name of default shell program*], does not exist. Please contact the system administrator.

The user's default shell program, for example `/bin/csh`, as listed in `/etc/passwd` can't be found.

You don't have permission to execute your default shell program, [*name of shell program*]. Please contact the system administrator.

The user doesn't have permission to execute the default shell program: (perhaps the system administrator made a directory in which the name differed from that in the `/etc/passwd` file).

Invalid user id [*ID number*]. Please contact the system administrator.

The user ID is not used in the `/etc/passwd` file.

Invalid group id [*group ID number*]. Please contact the system administrator.

The group ID is not used in the `/etc/passwd` file.

Your shell program, [*name of shell program*], is not a standard shell; thus, the session type will be Console Emulator.

There are three standard shells—`/bin/sh`, `/bin/csh`, and `/bin/ksh`. If the user's entry in `/etc/passwd` lists a different shell, this message is displayed in an alert box whenever the user chooses the Every Session or This Session Only button in the Change Session Type dialog box. To define the shell program a standard one used by your system, add it to `/etc/shells`.

The [*kind of session*] session startup program, [*name of startup program*], does not exist. A console emulator session will be started instead.

`/mac/bin/mac32`, or the chosen session type is missing.

You don't have permission to execute the [*name of session*] session startup program, [*name of program*]. A console emulator session will be started instead.

`/mac/bin/mac32`, or the chosen session type, is not executable by the user.

Your default shell program, [*name of default program*], does not exist.

`/bin/sh` will be used instead.

The default shell for root, for example `/bin/csh`, as listed in `/etc/passwd`, doesn't exist.

Overriding the default time zone

If you wish to display a time and date different from that used by the system, change the `TZ` environmental variable. For example, if you are on a business trip in France and log in from a terminal there, you will probably want the system to use the time and date of that time zone.

The value of `TZ` is kept in `/etc/timezone`. If your time zone differs from the one used by the system, then change it in your local shell setup files. See Chapter 3, “User and Group Administration,” for more information on these files.

The command you enter, however, depends on your command shell. For example, if you want to change to Pacific Daylight Time (in the U.S.) and are working in the C shell, put in your `.login` file

```
setenv TZ PST8PDT
```

Working in the Bourne or Korn shell, put in your `.profile` file

```
TZ=PST8PDT; export TZ
```

Errors while changing your password

Any of the following messages mean that password requirements are not met. These messages are displayed when the user attempts to click OK in the Change Password dialog box.

- **Your password must be at least six characters long.**
- **Your password must contain at least two alphabetic characters and one numeric or punctuation character.**
- **Your password cannot be a circular shift of your login name.**
- **Your new password must differ from your old one by at least three characters.**
- **This password can be changed only by the superuser.**
- **Sorry, your account has password aging restrictions. It has not been long enough since your password was last changed.**

This doesn't match your original entry. Please try again.

If the user retypes his or her password in the confirmation dialog box incorrectly, this message is displayed.

Another user is modifying the password file. Please try again later.

Only one user can change the password file at a time. Someone else may be editing it with `vipw` or the `passwd(1)` command.

Other miscellaneous issues

When starting the machine, you press the power button but the machine doesn't start.

Check the power cord to make sure that both ends are tightly plugged into the correct socket. (One end plugs into the power supply at the back of the system and the other into the electrical outlet.)

Make sure that the keyboard is plugged into the appropriate port with the appropriate cable.

The electricity may not be functioning in the outlet; try one that you know works. If you are using a power strip, check to be sure that it is turned on.

A Macintosh icon with an unhappy face appears on the screen, accompanied by a chiming sound.

Turn the machine off, using the switch on the back of the computer. Start up from a floppy disk containing system software to rule out any hardware problems. If the system boots from the floppy disk, reinstall the system software onto your MacPartition.

If the system fails to boot from the floppy disk, contact your authorized Apple dealer.

When you start the computer, a floppy disk icon with a blinking question mark shows up in the middle of the screen.

Make sure that the external hard disk is turned on.

Restart the computer, using the programmer's switch. The computer may not have recognized all of your disks during startup.

Examine the SCSI cables for proper configuration.

Make sure that the SCSI chain has been terminated properly.

Verify that each SCSI device has its own separate SCSI ID (0 to 6).

System software may not be installed. Install, or reinstall, if necessary.

Boot blocks may be damaged. Reinstall system software.

A system file may be corrupt. Reinstall system software.

The internal or external disk, or both, may have crashed. After trying all of the other suggestions, contact your authorized Apple dealer.

After double-clicking the A/UX Startup icon, you receive the error message

`Chroot failed.`

If you have a different device for the root A/UX file system than the device that contains A/UX Startup, choose Preferences from the General menu and change the `(default)/` field to reflect the SCSI ID number for your root file system. Click OK and choose Quit from the File menu. Then double-click the A/UX Startup icon and attempt to bring up A/UX again.

◆ **Note** The only reason to change from the `(default)/` SCSI is when A/UX Startup and A/UX are not on the same SCSI device, or when you boot the system with A/UX Startup on a floppy disk. ◆

During the launching of A/UX, `fsck` locates a problem with the file system. You are asked to click the Repair button.

Let `fsck` automatically repair the file system by clicking Repair.

Another method is to run `fsck` from A/UX Startup. Reboot your system. Cancel the boot process to enter the A/UX Startup command shell window. Enter

```
fsck -p /dev/dsk/cxd0s0
```

where *x* stands for the SCSI number of your device.

See Chapter 8, “Checking the A/UX File System: `fsck`,” for more information on repairing your file system.

If these suggestions don't work, contact your local authorized Apple dealer for assistance.

A/UX is frozen at the Login dialog box and the keyboard does not respond.

Make sure that the keyboard is plugged into the back of the Macintosh with the appropriate cable.

Restart the machine and cancel the start up process, so that you are in A/UX Startup. Use the `cat` command to display the `/etc/inittab` file; check that all of the `inittab` settings are correct. If you are using NIS, make sure that the server is running.

The error message `fserr: filesystem full` appears on the screen every few minutes.

The file system has run out of space or inodes. Enter `df`, which displays the number of blocks and inodes available for use on the current file system. To free space and inodes, make backups of old files to be removed, either on tape or on floppy disks, and then remove them from the full file system. If you log in as root, the messages may stop while you clear out the files.

While you try to partition a disk for A/UX with Apple Hard Disk SC setup, the drive cannot be found.

Make sure that the external hard disk is turned on.

Make sure that the SCSI cables are properly configured.

Make sure that the SCSI chain has been terminated properly.

If the disk is not an Apple product, contact your dealer or manufacturer to get the right software.

A non-SCSI device, such as the Apple Hard Disk 20, cannot be read by Apple Hard Disk SC setup.

Two disks with the same SCSI ID may be turned on. Shut the machine down and turn one of the drives off. Insert the point of a pushpin or a straightened paper clip into the small hole in the SCSI selector switch to change the SCSI ID. Turn on the drive and the Macintosh computer.

The drive may be damaged. Contact your authorized Apple dealer.

The error message `m_expand returning 0` appears on the screen every few minutes.

Increase the number of `NMBUFS` with the `kconfig` command. `NMBUFS` allocates buffers for networking; when installing NFS, the number should be increased. Remember that these changes do not take effect until the kernel has been rebooted.

The error message `file:table is full` appears on the screen every few minutes.

The system file table is full and needs to be increased. The `kconfig` command allows the `NFILE` parameter to enlarge the table. When you increase the `NFILE` parameter, the `NINODE` parameter should be equal to or greater than the `NFILE` parameter. (They are usually kept at the same number.) The total memory configuration of your system should determine the size of your `NFILE` and `NINODE` parameters.

The error message `proc:table is full` **appears on the screen every few minutes.**

The system has attempted to increase the total number of system processes beyond the default number set in the kernel. The `kconfig` command allows the `NPROC` parameter to reflect a higher number. Increase the `NPROC` parameter in increments of 25 until the message no longer appears. You must reboot each time you change the kernel with the `kconfig` command.

While you are using the `tar` **or** `cpio` **command with a floppy disk, the error message** `cannot open /dev/floppy0` **is displayed.**

The drive that has the floppy disk could be `/dev/floppy1`, or else the disk is write-protected.

When you are using `chgrp` **and** `chown`, **the error message** `filename: Not owner` **appears.**

You do not have the appropriate permissions to change owner or group of that file. Enter `su` to become root and run the command again.

While trying to unmount a mounted file system, you encounter the error message: `/filesystem: Device busy.`

The file system *filesystem* is currently in use. Change to the root directory by entering `cd /`

Enter the `umount` command again. Make sure that no other windows are open in which users have changed directories to the file system you wish to unmount.

Somebody else on the network may be accessing that directory. Use the `who` command to see if someone else is using that file system.

While you are using `tar` or `cpio` with the Apple Tape Backup unit, an error message appears indicating that the utility cannot open `/dev/rmt/tcx`.

The tape is write-protected.

The device file you selected was assigned an incorrect SCSI ID number. Reselect it with the correct device number.

The kernel may not have been updated with the correct drivers. Verify by running the `module_dump /unix` command. Look for the `tc` driver in the list. If it isn't there, run `autoconfig` to configure the kernel with the tape driver. Reboot the computer.

When using `tar`, you failed to use `-f /dev/rmt/tcx`, where *x* is the SCSI number.

While you are creating a file system on a disk that has been initialized and partitioned with A/UX, an error message is displayed indicating that the block limit is too large to fit on that partition.

Check the slice number to be sure that it coincides with the partition you gave it while using Apple Hard Disk SC Setup.

Perhaps the number of blocks that you specified while using `mkfs` for a SVFS file system is too large.

Make sure that you used the correct SCSI ID number.

Files sent to the printer have not printed.

Make sure that the printer is not out of paper.

Make sure that the cable from the LaserWriter to the Macintosh is plugged in to the right ports.

If using the `lpr` spooler, run `lpc status` to verify that the printer is accepting requests.

With the `lp` spooler, enter the `lpstat` command. Restart the scheduler with the command `/usr/lib/lpsched`.

If using `lpr` and the default printer destination, make sure a printer has been chosen with the Chooser.

Index

A

- A/UX Autorecovery partition 4-18
- A/UX file systems 4-6 to 4-7
 - overview 8-3 to 8-13
- A/UX Finder
 - modes 3-6 to 3-7
- A/UX kernel 10-3
- A/UX Startup program 2-4 to 2-6
 - accessing A/UX file system 10-4 to 10-5
 - automatic boot 2-11
 - changing prompt 2-5
 - commands 2-6 to 2-7
 - loading A/UX kernel 10-3
 - menus 2-8 to 2-12
 - quitting 2-8
 - using for troubleshooting 2-7
- A/UX 3.0 CD-ROM*
 - using `cpio` 5-14
- A/UX won't boot 6-3 to 6-4
- accept C-4
- accounting D-1 to D-25
 - reports D-6 to D-15
- acctcom command D-6, D-12 to D-14
 - options D-13 to D-14
- acctdusg program D-9
- acctwtmp program D-16
- adding a user, manually 3-14 to 3-18

- adduser program
 - and setup files 3-7
 - default shell for 3-6
- adm administrative group 1-9
- admin administrative login 1-9
- administrative groups 1-9
- administrative logins 1-8
- Apple Hard Disk SC setup
 - troubleshooting of 10-14
- Apple Personal Modem
 - dial-in access 7-33
 - dial-out access 7-30 to 7-32
 - setup differences 7-33
- AppleTalk print queue 7-6
- Apple Tape Backup 40SC drive
 - error messages 10-16
 - software 5-9
- archival utilities. *See* `dump.bsd`,
`cpio`, `tar`, `pax`
- archives
 - compressing 6-14 to 6-15
 - definition of 5-2
- autoconfig program 2-15
- autolaunch variable 2-11
- automating routine tasks 5-45 to 5-48
- autorecovery facility 6-3 to 6-11
 - administration 6-6
 - command line 6-5

- how it works 6-5
- lock file 6-16
- messages at boot time 6-7 to 6-8
- autorecovery file system 6-2
 - checking for integrity 6-8 to 6-9
- Autorecovery partition 4-18
- autorecovery program
 - cluster number 2-12

B

- background processes, starting 2-17
- backing up 5-1
 - media for 5-7 to 5-10
 - networked file systems 5-7
 - over a network 5-29
 - reasons for 5-1
 - run-level considerations 5-7
 - strategies for 5-3 to 5-4
 - using `cpio` 5-11 to 5-17
 - using `dump.bsd` 5-29 to 5-34
 - using `pax` 5-10, 5-40 to 5-43
 - using `tar` 5-20 to 5-28
 - utilities for 5-10 to 5-11
 - verifying media 5-43 to 5-44
- backup devices
 - `cpio` filter requirement 5-13
 - `dd` block size 5-43 to 5-44
 - `dump.bsd` blocking factor 5-31

- backup devices (*continued*)
 - tar blocking factor 5-22
 - tar default 5-21
- backup media
 - capacity, in blocks 5-22
 - capacity, in bytes 5-8
 - listing contents with cpio 5-16 to 5-17
 - listing contents with pax 5-42
 - listing contents with tar 5-26
- baud rate 7-20
- bcheckrc program. *See* /etc/bcheckrc program
- bin administrative group 1-9
- bin administrative login 1-8
- block devices 5-4, 8-12
- blocking arguments
 - cpio filters 5-13 to 5-14
 - dd 5-44
 - dump.bsd 5-31
 - tar 5-22
- block zero block (BZB) E-8
- boot command
 - A/UX dialog box 2-10
 - from A/UX Startup 2-7
- booting
 - phases of 2-13 to 2-17
 - troubleshooting 2-13
- Booting dialog box 2-10 to 2-11
- Bourne shell 3-5, 3-6, 3-8
- buffer activity report 9-10
- buffer cache 8-10

C

- capacity arguments
 - dump.bsd 5-31
 - tar 5-22
- capacity, media 5-8
- catastrophic failure
 - restoring after 5-36 to 5-38
 - what to do 6-3 to 6-5
- ccat utility 6-15

- CD-ROM
 - accessing man pages 6-17 to 6-19
 - mounting A/UX file systems 6-17
- character devices 5-4
- chargefee procedure D-16
- chgrp command 3-18
- child process 9-11
- chmod command 3-18, 3-29 to 3-32
- chsh command 3-23 to 3-24
- ckpacct procedure D-16 to D-17
- CML (configuration master list) 6-4
 - updating 6-4 to 6-6
- command shell, definition 3-5
- CommandShell, definition 3-5
- command usage reports D-9 to D-11
- compact utility 6-14
- compressing files 6-14 to 6-15
- compress utility 6-14
- configuration master list. *See* CML
- copying files by dragging 5-11
- copying with the Finder 5-11
- copy utilities 5-10
- core files 6-12
- corrupted accounting files D-23 to D-24
- cpio command 5-11 to 5-19
 - and *A/UX 3.0 CD-ROM* 5-14
 - advantages of 5-11
 - backing up 5-14 to 5-16
 - block option 5-15
 - cannot open device 10-15
 - compatibility option 5-15
 - compressing file archives 6-14 to 6-15
 - copy in option 5-17
 - copy out option 5-15
 - device option 5-14
 - directory option 5-17
 - disadvantages of 5-12
 - filters required 5-13
 - listing contents 5-16 to 5-17
 - modification option 5-17
 - moving a user across file systems 3-21 to 3-22
 - options 5-12

- recovering files 5-17 to 5-19
- recovering multiple files 5-19
- recovering specific files 5-17
- restoring missing files 8-59
- table of contents option 5-17
- and tape backups 5-13
- unblocking 5-17
- update option 5-17
- verbose option 5-15, 5-17
- CPU activity report 9-7, 9-8 to 9-9
- CPU counter 9-3
- critical system files 6-4. *See also* CML
- cron.allow file 5-45 to 5-46
- cron daemon 5-45
- cron.deny file 5-45 to 5-46
- crontab command 5-48
- crontab file format 5-46
- cron utility 5-46
- C Shell 3-5, 3-7
 - setup files 3-6, 3-7
- current directory 3-5

D

- daemon administrative group 1-9
- daemon administrative login 1-8
- data blocks 8-14
 - file system updates of 8-14
- data partition map entry (DPME) E-7
- date command 1-7
- dd command 5-43 to 5-44
 - blocking arguments for 5-44
- default backup devices
 - dump.bsd 5-30
 - tar 5-21
- (default)/ parameter for SCSI ID 2-22, 10-12
- default shell program 3-5 to 3-6
 - problems with 10-8
- default startup program, changing 3-23 to 3-24
- Details window 4-25
- /dev directory 5-4, 8-10 to 8-11
- device activity report 9-13 to 9-15

- device drivers 8-12
 - definition 7-3
 - initializing 2-15
- device files 5-4 to 5-6, 7-29
 - standard for A/UX 5-5
 - used by lpr 7-5
- devices 7-6
 - /dev/modem file 7-4
 - /dev/printer file 7-4
- df command 4-7, 4-8
 - displaying blocks and inodes 10-13
- dial-in access 7-32 to 7-33
- direct blocks 8-6
- directories, as lists of inodes 8-4 to 8-5
- directories, moving 3-21
- directory hierarchies 4-6
- disk drive names 4-29 to 4-30
- disk drive parameters 4-30
- disk partitions. *See* partitioning hard disks, partitions
- disk partition map (DPM) 4-4, 4-25
- disk space, reclaiming 6-11 to 6-16
- disks, preparing for use 4-2 to 4-3
- disks. *See* floppy disks; hard disks, SCSI
- dodisk procedure D-15, D-17
- dp utility 4-11, E-2 to E-9
- dump.bsd command 5-29 to 5-30, 5-32 to 5-34
 - advantages of 5-29
 - backing up 5-32 to 5-34
 - blocking arguments 5-31
 - capacity arguments 5-31
 - default backup device 5-30
 - default dump level 5-31
 - definition of 5-29
 - disadvantages of 5-29
 - dump levels 5-30 to 5-31
 - full system backups 5-32 to 5-33
 - how it works 5-32
- dumpdates file 5-30
- dump frequency 5-34
- dump levels
 - monthly backup strategy 5-30
 - with dump.bsd 5-30

E

- error messages 10-2
 - acctg already run D-21
 - BAD BLK 8-34
 - BAD BLKS 8-55
 - BAD/DUP DIR 8-48
 - BAD/DUP FILE 8-47
 - BAD FREEBLK COUNT 8-55
 - BAD FREE LIST 8-55
 - BAD i 8-29
 - BAD INODE 8-36, 8-44
 - BAD RETURN 8-35
 - BAD STATE 8-30, 8-36
 - BAD SUPER BLOCK 8-27
 - Bad -t option 8-23
 - BLK(S) MISSING 8-53, 8-55
 - cannot alloc 8-23
 - cannot examine spool directory 7-16
 - CANNOT READ 8-24
 - CANNOT SEEK 8-24
 - CANNOT WRITE 8-24, 8-25
 - can't create 8-25
 - can't fstat 8-25
 - can't get memory 8-25
 - Can't lock cml file 6-10
 - can't make sense 8-26
 - can't open 8-26
 - can't stat 8-26
 - CG BAD MAGIC 8-53
 - connect acctg failed D-22
 - couldn't start printer 7-16
 - CPG OUT OF RANGE 8-27
 - Device busy 4-37, 10-15
 - device previously named 6-10
 - DIR CONNECTED 8-44
 - DIRECTORY CORRUPTED 8-36
 - DIRECTORY LENGTH 8-36, 8-44
 - DIRECTORY MISALIGNED 8-30
 - DIRECTORY TOO SHORT 8-37
 - DIR I CONNECTED 8-44
 - DUP BLKS 8-55
 - DUP I 8-30, 34
 - DUPS/BAD I 8-37, 8-38
 - DUPS/BAD IN ROOT INODE 8-37
 - DUP TABLE OVERFLOW 8-30
 - ERROR: acctg already run D-21
 - ERROR: connect acctg failed D-22
 - ERROR: invalid state D-22
 - ERROR: locks found D-22
 - ERROR: Spacct? D-22
 - ERROR: turnacct switch D-22
 - ERROR: wttmp already exists D-22
 - ERROR: wttmpfix errors D-22
 - esch: no consistent type 6-8
 - /etc/dumpdates: No such file or directory 5-30
 - EXCESSIVE BAD BLKS 8-56
 - EXCESSIVE BAD BLOCKS 8-31
 - EXCESSIVE DUP BLKS 8-31, 8-56
 - EXTRA 8-38, 8-39
 - EXTRANEIOUS HARD LINK 8-39
 - filename* was not replaceable 6-9
 - FREE BLK COUNT 8-56
 - FREE BLK COUNT(S) 8-53
 - FREE INODE COUNT 8-48
 - IMPOSSIBLE MINFREE 8-27
 - Incompatible options 8-27
 - Inconsistent mount times 6-7
 - INCORRECT BLOCK COUNT 8-31
 - Invalid -s argument 8-27
 - invalid state D-22
 - is not a block 8-28
 - jobs queued, but 7-13
 - LINK COUNT 8-32, 8-48, 8-49
 - locks found D-22
 - lost+found IS NOT A DIRECTORY 8-45, 8-49

error messages (*continued*)

- lpr:jobs queued, but 7-13
- lprm:cannot restart
 - printer daemon 7-15
- lpr:printer queue is disabled 7-14
- lpr:unknown printer 7-13
- MAGIC NUMBER WRONG 8-27
- MISSING 8-39, 8-40
- NAME TOO LONG 8-40
- NCG OUT OF RANGE 8-27
- MCYL DOES NOT JIVE 8-27
- no consistent type 6-8
- no daemon present 7-15
- NO lost+found DIRECTORY 8-45, 8-50
- NO SPACE LEFT IN
 - lost+found 8-46, 8-50, 8-51
- NO WRITE 8-28
- option? 8-23
- OUT OF RANGE 8-44
- PARTIALLY ALLOCATED INODE 8-32
- PARTIALLY TRUNCATED 8-32
- POSSIBLE FILE 8-33
- previously pnamed 6-10
- printer is down 7-15
- printer queue is disabled 7-14
- ROOT INODE 8-41, 8-42
- sending to host 7-15
- Size check 8-28
- SIZE PREPOSTEROUSLY LARGE 8-27
- SORRY. No lost+found 8-46, 8-51
- SORRY. NO SPACE 8-46, 8-51
- Spacct? D-22
- SUMMARY INFORMATION BAD 8-54
- tcb: No such device 5-13
- TRASHED VALUES 8-27
- turnacct switch D-22
- UNALLOCATED I 8-42 to 8-43
- UNDEFINED OPTIMIZATION 8-29
- UNKNOWN FILE TYPE 8-33
- unknown printer 7-13
- UNREF DIR 8-47, 8-51
- UNREF FILE 8-52
- waiting for host 7-15
- waiting for printer 7-14
- Warning. Inconsistent mount times 6-7
- Warning: printer is down 7-15
- Warning: no daemon present 7-15
- wtmpfix errors D-22
- ZERO LENGTH 8-43
- errors
 - A/UX won't boot 6-3 to 6-4
 - booting 6-3 to 6-4
 - hard I/O 5-19
- eschatology 6-3
- escher utility 6-6
- /etc/bcheckrc program 2-16
- /etc/cml/init2files. *See* CML
- /etc/getty file 7-19
- /etc/gettydefs file. *See* gettydefs file
- /etc/group file 3-24 to 3-27
- reading at startup 3-9
- troubleshooting 10-7
- /etc/inittab file. *See also* inittab file
- action field 7-20 to 7-21
- id field 7-20
- run-level field 7-20
- troubleshooting 10-13
- /etc/macsysinit file
- launching Macintosh environment 2-14
- /etc/motd 1-6
- /etc/passwd file
- components defined 3-10 to 3-11
- creating an entry 3-10 to 3-11
- troubleshooting 10-7
- /etc/profile file 3-6, 3-8
- /etc/rc file 2-17, 7-19
- /etc/sysinitrc shell program 2-15
- /etc/termcap file 7-26
- eupdate utility 6-7
- eu utility 6-6

F

- file access report 9-15 to 9-16
- file-access sequence 4-7
- /FILES 1-19
- files
 - backing up with cpio 5-14 to 5-16
 - backing up with pax 5-40 to 5-43
 - backing up with tar 5-20 to 5-28
 - compressing 6-14 to 6-15
 - copying by dragging 5-11
 - decompressing 6-15
 - how to get inodes for 8-4 to 8-5
 - listing with restore 5-34 to 5-35
 - recovering with cpio 5-17 to 5-19
 - recovering with pax 5-42 to 5-43
 - recovering with restore 5-34 to 5-39
 - recovering with tar 5-26 to 5-28
 - removing unneeded 6-12
 - trimming size of 6-13 to 6-14
 - truncating 6-14
- file systems
 - checking. *See* fsck command
 - comparison 4-29
 - definition of 4-6
 - fragmentation 8-19
 - inconsistencies 8-13
 - listing file system commands 1-18
 - making 4-29 to 4-32
 - mounting 4-32 to 4-36
 - optimizing performance 8-19
 - overview of A/UX 8-2 to 8-13 and partitions 4-6
 - restoring full backups 5-36 to 5-38
 - unmounting 4-36 to 4-37
 - updates 8-14 to 8-15

- file table full error message 10-14
- file table is full error message 1-17
- find command 1-4
 - mtime option 5-3
- Finder
 - modes 3-6 to 3-7
- floppy disks
 - as media for backup 5-7 to 5-8
 - ejecting at launch 2-11
 - media life 5-4
- floppy drives
 - eject devices 5-22
 - special features 5-6
- fragmentation 8-19
 - reporting 8-57
- free list 8-15, 8-54 to 8-57
 - file system updates of 8-15
- fsck errors. *See also* error messages
 - initialization 8-23 to 8-29
 - options 8-24 to 8-29
 - Phase 1 8-29 to 8-33
 - Phase 2 8-33 to 8-43
 - Phase 3 8-44 to 8-47
 - Phase 4 8-47 to 8-52
 - Phase 5 (SVFS) 8-54 to 8-56
 - Phase 5 (UFS) 8-53 to 8-54
 - Phase 6 (SVFS) 8-57
- fsck utility 8-1 to 8-58
 - checking autorecovery 6-8
 - cleanup functions 8-57
 - dialog box 2-17
 - error messages 8-23 to 8-57
 - examples 8-18 to 8-20
 - finding problems during launch 10-13
 - fixing errors on disks 10-4
 - how it works 8-13
 - six phases of 8-15 to 8-17
 - when to use 8-17 to 8-18
 - with autorecovery 6-5
- fsentry command 4-33 to 4-34
- fstab file
 - creating entries 4-32

- dump frequency 5-34
- fields for automatic file system check 8-20 to 8-22
- reference 8-17
- full backups 5-3
 - using `dump.bsd` 5-32 to 5-34

G

- General dialog box 2-11
- gettydefs file 7-19 to 7-21
- getty file. *See* `/etc/getty` file
- getty process 7-20 to 7-21, 7-24, 7-28
 - determining settings of 7-20 to 7-21
- GID 3-9
 - out of range 10-9
- graceful shutdown 10-7
- Greenwich Mean Time 1-6
- group ID 3-9
 - creating 3-24 to 3-25
 - out of range 10-9
- groups, A/UX maximum 3-27
- Guest account. *See* Guest user
- Guest user
 - account for 1-7
 - security on 3-10

H

- Hard Disk SC Setup 4-11
 - general description 4-9 to 4-10
 - troubleshooting 10-14
- hard disks, SCSI
 - reinitializing error-prone 10-5 to 10-6
- hard I/O errors, while backing up 5-19, 5-44
- help command, A/UX Startup 2-8
- holidays
 - updating D-5 to D-6
 - file, format of D-6
- home directory 1-8, 3-5
 - problems with 10-8
- HOME variable
 - built-in 2-12
- host name 1-5, 3-7

I, J

- ImageWriter print queue 7-6
- in-core blocks 8-14
- incremental backups 5-3
 - using `cpio` 5-16
 - using `dump.bsd` 5-29 to 5-30, 5-33
- indirect blocks 8-6 to 8-7, 8-14
 - file system updates of 8-14
- initgroups 3-9
- initialization script 10-11
- initial processes 7-20
- init program, considerations when running 1-12 to 1-13
- inittab file
 - changing for a new terminal 7-18 to 7-19
- init2files. *See* CML
- inodes
 - access time 8-7
 - definition 8-4
 - file system updates of 8-14
 - how to get file names for 8-5
 - location 8-4
 - modification time 8-7
- inode table is full error message 1-17

K

- kconfig command 1-17, 10-14, 10-15
 - NPROC parameter 10-15
- kernel
 - loading 2-14
 - rebuilding 2-15
- key system files. *See also* CML 6-4
- known parameters 8-4
- Korn shell 3-6, 3-7, 3-8
 - `.kshrc` setup file 3-6

L

- lock files
 - autorecovery 6-10
 - system accounting D-19

- logging in 3-4
 - establishing user's environment 3-9
- logging out 2-18
- logical blocks 8-4
- .login file 3-6
- login files 3-9
- login name 3-4
- login program 3-9
- login shell 3-7 to 3-8
- lost+found directory 4-31
- lp
 - accepting requests C-15
 - canceling requests C-16
 - determining status C-5
 - disable command C-16
 - enable command C-17
 - rejecting requests C-15
- lp administrative group 1-9
- lp administrative login 1-8
- lp print spooler C-1 to C-23
 - commands for general use C-3
 - commands for lp administrator C-3
 - configuring the system C-8 to C-13
 - handling requests C-15 to C-18
 - syntax of command C-14
 - system files C-21 to C-22
 - troubleshooting C-18 to C-20
- lpadmin command C-4, C-8
- lpc commands
 - abort 7-10
 - disable 7-11
 - down 7-12
 - enable 7-11
 - start 7-10
 - stop 7-11
 - up 7-11
- lpc error messages 7-16
- lpd error messages 7-16
- lpd print scheduler 7-5
- lpmove C-4
- lpq command 7-10
 - error messages 7-14 to 7-15
- lpr error messages 7-13 to 7-14

- lprm command 7-10
 - error messages 7-15
- lpr print spooler 7-4 to 7-17
 - commands for general use 7-10
 - commands for lpr administrator 7-10 to 7-12
 - setting up 7-6 to 7-9
 - troubleshooting 7-13 to 7-16
- lpsched C-4
- lp scheduler C-5 to C-7
 - starting C-6 to C-7
 - stopping C-6 to C-7
- lpshut C-4
- lpstat command C-5
- ls command
 - using with cpio 5-12
- /mac/bin/mac32 command 3-9

M

- Macintosh Operating System
 - bypassing for A/UX 2-4
- Macintosh volume 4-10
- MacPartition 4-4, 4-5
- macsysinit file. *See*
/etc/macsysinit file
- MacTerminal application 7-21
- mail administrative group 1-9
- man pages
 - on CD-ROM 6-17 to 6-19
- media life 5-4
- media size 5-8
- memory size, and swap space 4-26 to 4-27
- message and semaphore activity report 9-13
- message of the day 1-6
- Misc UNIX partition type 4-18, E-1
- mkfs command 4-31 to 4-32, 10-16
- modems 7-30 to 7-35
 - Apple Personal Modem 7-33
 - dial-in access 7-32 to 7-33
 - dialing out 7-34 to 7-35
 - dial-out access 7-30 to 7-32
 - setting up 7-30

- monacct procedure D-9, D-17 to D-18
- mount command 4-7, 4-34 to 4-36
 - accessing partitions 4-7 to 4-8
 - example of 4-8
 - for floppy disks 4-35
 - for hard disks 4-35
- mounting
 - automatically 4-33 to 4-34
 - temporarily 4-34 to 4-36
- mount points 2-16
 - creating 4-34 to 4-35
 - definition of 4-7
- mt command 5-6
 - example 5-19
- multiple archives 5-6
 - backing up with cpio 5-16 to 5-17
 - backing up with tar 5-24 to 5-25
 - extracting 5-19
 - extracting with tar 5-28
 - positioning 5-19
- multi-user mode 1-11

N

- naming the system 1-5
- network backups 5-29
- network communication 1-8
- networked file systems, backing up 5-7
- newconfig program 1-15 to 1-16
- newfs command 4-30 to 4-31
- newgrp command 3-27
- NFILE parameter 1-17, 10-14
- NINODE parameter 1-17, 10-14
- NIS
 - passwords 3-10
 - with adduser program 3-12
- NMBUFS 10-14
- nobody administrative login 1-8
- no-rewind devices 5-6
- NPROC 1-17
- nuucp administrative login 1-8

O

- optimizing performance 8-19

P

- pack utility 6-14
- parent process 9-11
- partitioning
 - and file sharing 4-13
 - definition 4-1
- partitioning hard disks. *See* partitions
 - troubleshooting 10-14
- partition map 4-4, 4-25
- partition names
 - eliminating duplicate E-5
 - used by A/UX utilities 4-11
- partitions
 - adding 4-15 to 4-21
 - checking information about 4-24 to 4-26
 - definition of 4-3
 - grouping of free space 4-21 to 4-22
 - moving 4-12, 4-23 to 4-25
 - predefined 4-16
 - referring to 4-11
 - removing 4-13 to 4-15
 - types of 4-18
- password aging 3-39
- password program 3-33
- passwords
 - incorrect 10-8
 - permissions for 3-32
 - protecting A/UX Startup 2-12
 - requirements 10-10
 - restrictions for System V 3-39
- pax command 5-2, 5-40 to 5-43
 - advantages of 5-40
 - backing up with 5-41 to 5-42
 - disadvantages of 5-40
 - link option 5-41 to 5-42
 - listing contents 5-42
 - read option 5-42
 - recovering files 5-42, 5-43
 - restoring missing files 8-58 to 8-59
 - write option 5-41
- pcat utility 6-15
- peripheral devices 7-1

- permissions 3-4
 - directory 3-27, 3-29
 - folder 3-27, 3-29
- physical blocks 8-3
- pname utility 6-7
- ports 7-3 to 7-4
- powering down. *See* shutting down
- prctmp procedure D-15
- prdaily procedure D-6
- printcap database 7-6 to 7-9
- printer interface program 7-6
- printer output filters, writing 7-16 to 7-17
- printer queues, access to 7-9
- printers
 - default 7-7
 - naming 7-7
 - serial-line 7-7 to 7-8
 - spool directory 7-8
 - system default destination 7-7
- printers, non-apple 7-6 to 7-7
- printing, troubleshooting 10-16
- print job request 7-5
- print spooler 7-4 to 7-5
 - modifying 7-6
- proc table full error message 1-17, 10-15
- .profile file 3-8
- prompt 3-7
- prtacct procedure D-14 to D-15
- pswitch counter 9-11
- ptecms.awk file D-7
- ptelus.awk file D-7
- pwck command 3-38

Q

- queue activity report 9-12 to 9-13

R

- raw devices 5-4
- rc file. *See* /etc/rc
- rdump command 5-29
- recovering files
 - using cpio 5-17 to 5-19

- using pax 5-42 to 5-43
- using restore 5-34 to 5-39
- using tar 5-26 to 5-28

- redundancies 8-4
- references, additional about UNIX B-2
- reject C-4
- remove procedure D-16
- remsh command 7-9
- Repair button in fsck dialog box 10-13
- restarting A/UX 2-18
- Restart menu item 2-10
- restore command
 - advantages of 5-29
 - disadvantages of 5-29
 - interactive mode 5-38 to 5-39
 - listing files 5-34 to 5-35
 - restoring files 5-35
 - restoring full backups 5-36 to 5-38
 - restoring missing inodes 8-58
 - and tape backups 5-31
- restoring missing files
 - from CD 8-59
 - using cpio 8-59
 - using pax 8-60
 - using restore 8-59
- restricted shell 3-24
- rfloppy. *See* raw devices
- root account 1-3
 - privileges 1-10
- root administrative group 1-9
- root administrative login 1-8
- root file system 2-15
- rsh command 3-24, 7-9
- runacct procedure D-12 to D-13, D-18 to D-24
 - error messages D-21 to D-22
 - failure of D-19 to D-21
 - fixing corrupted files D-23 to D-24
 - restarting D-18 to D-19
 - status D-18
- run levels
 - changing 1-12 to 1-13
 - default in /etc/inittab 7-19

S

- sa1 command 9-4
- sa2 command 9-4
- sadc command 9-3, 9-8
- sag command 9-6, 9-18
- sar command 9-7 to 9-8
 - reports 9-9 to 9-16, 9-17
 - syntax 9-7
- screen, frozen 10-13
- SCSI, definition of 4-2
- selective backups 5-3
- serial ports 7-3 to 7-4
 - setting up 7-28 to 7-29
- session type, invalid 10-9
- setgid 3-32
- setport command 7-29
- setport Commando dialog box 7-28
- setuid 3-32
- shell programs
 - and setup files 3-6 to 3-8
 - changing default program 3-23
 - default 3-5
 - selecting for new user 3-14
- shutacct command D-25
- shutdown message, sending 2-20
- shutting down
 - from A/UX command line 2-18 to 2-21
 - from the Finder 2-18
- single-command activity report 9-17 to 9-18
- single-user mode 1-10
 - from multi-user mode at shutdown 2-18 to 2-19
 - making default mode 1-10
- size, media 5-8
- slice 30 4-6. *See also* MacPartition
- slice 31 4-6
- slice numbers 4-4
 - Apple conventions for 4-6
 - assigning permanent E-6 to E-9
- special files 8-10
- spooler system 7-14

- starting up the system 2-4 to 2-17
 - overview 2-2, 2-3
- startup device
 - changing 2-22 to 2-23
 - order of 2-21
- startup error messages 8-23
- sticky bit 3-34, 9-11
- superblocks
 - definition of 4-7
 - file system updates of 8-14 to 8-15
 - list of contents 8-9
 - SVFS vs. UFS 8-9
- superuser. *See* root account
- swap device 9-11
- swapping activity report 9-11
- swap space
 - adding 4-26 to 4-28
 - and memory size 4-26, 4-27 to 4-28
- switching activity report 9-11
- sync command 8-14
- sys administrative group 1-9
- sys administrative login 1-8
- sysinitrc script. *See* /etc/sysinitrc script
- syslog file 7-16
- system accounting package D-1 to D-21
 - error messages D-21 to D-22
 - starting D-3 to D-4
 - stopping D-25
- system activity package 9-1 to 9-18
 - counters 9-3
 - data collector 9-2, 9-6
 - functions, setting up 9-5 to 9-6
 - graph command 9-18
 - graph, printing requirements of 9-18
 - reports 9-6 to 9-8
 - single-command activity report 9-17 to 9-18
- system call activity report 9-11 to 9-12
- system console, displaying during shutdown 2-19
- system files
 - backing up 1-3
 - monitoring growth 6-13

- System Folder
 - creating 3-8
 - personal 3-8
- system startup 2-4 to 2-17
 - overview 2-2, 2-3
- system time
 - adjusting for daylight saving 1-6
 - moving to another time zone 1-7
 - overriding 10-10
 - reasons for resetting 1-6

T

- table overflow status 9-15
- tacct files
 - list of D-15
 - maintaining integrity of D-24
- tail utility 6-13
- tape backups
 - using cpio 5-14
 - using dump.bsd 5-31
 - using pax 5-40
 - using tar 5-23
 - when to use 5-9
- tape capacity 5-8
- tape cartridges. *See also* backing up size 5-8
- tape controller 5-13 to 5-14
- tape drives
 - no-rewind devices 5-6
 - special features 5-6
- tar command 5-20 to 5-28
 - advantages of 5-20
 - backing up 5-23 to 5-26
 - block option 5-22
 - cannot open device error message 10-15
 - capacity option 5-22
 - create option 5-23
 - default backup device 5-21
 - device option 5-23
 - disadvantages of 5-20
 - listing contents 5-26
 - multiple volume backup 5-21

- recovering files 5-26 to 5-28
- recovering latest version 5-27
- table of contents option 5-26
- and tape backups 5-21
- tcb filter 5-13
- termcap file 7-26
- terminals
 - attaching 7-21, 7-27
 - attaching a Macintosh computer as 7-21 to 7-25
 - attaching non-VT100 7-25
 - attaching VT100 7-25
 - using another computer as 7-21
- testing a hard drive 10-6
- time. *See* system time
- timex command 9-6, 9-17 to 9-18
- troubleshooting 10-1 to 10-16
 - autorecovery 6-7 to 6-11
 - print spooler 7-13 to 7-16
 - problems at startup 10-11 to 10-13
 - useraccount problems 10-7 to 10-9
- turnacct command D-16, D-22
- TZ environmental variable 10-10

U

UFS

- advantages over SVFS 4-29

UID (user ID)

- defined 3-4

- finding unused one 3-13

- invalid number error message 10-8

- read at login 3-9

- umask command 3-34 to 3-35

- umount command 4-36 to 4-37, 10-15

- UNIX reference materials B-2

- unmounting file system, problems with 10-15

- user identification number. *See* UID

users

- adding manually 3-14 to 3-18

- adding with `adduser` 3-13 to 3-14

- moving 3-21 to 3-22

- removing 3-19 to 3-20

- user's working environment

- planning 3-12 to 3-14

- specifying 3-19 to 3-20

- `/usr/lib/skel` file 3-6, 3-17

- uucp administrative group 1-9

- uucp administrative login 1-8

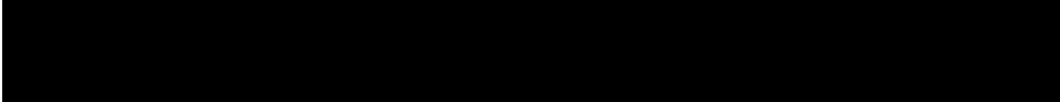
- UUCP communications package 1-8

V, W, X, Y, Z

- `vipw` command 3-15

- `wtmpfix` program D-20

- `zcat` utility 6-15



Appendix A: Files Unique to A/UX

This appendix list those commands and setup files found in A/UX Release 3.0 that are not part of standard UNIX distributions.

<code>/.aux_prefs</code>	preferred session type information
<code>/.cmdshellprefs</code>	CommandShell preferences data
<code>/.desk</code>	directory of Macintosh desktop files, by hostname
<code>/.desk/localhost</code>	Macintosh desktop file for host named “localhost”
<code>/.mac/localhost/Desktop Folder</code>	directory where “things on the Desktop” are put
<code>/.mac/localhost/Trash</code>	directory where “things that are thrown away” are put
<code>/FILES</code>	description data base for all files as shipped (<code>cm1(4)</code>)
<code>/bin/diskformat</code>	binary executable for <code>diskformat(1M)</code> —dependent format operation
<code>/bin/dp</code>	binary executable for <code>dp(1M)</code> —perform disk partitioning
<code>/bin/eject</code>	binary executable for <code>eject(1)</code> —eject diskette from drive
<code>/bin/terr</code>	binary executable for <code>terr(1M)</code> —turn on/off the reporting of extended errors
<code>/bin/pname</code>	binary executable for <code>pname(1M)</code> —associate named partitions with device nodes
<code>/bin/tp</code>	binary executable for <code>tp(1)</code> —manipulate tape archive
<code>/bin/version</code>	binary executable for <code>version(1)</code> —reports version number of files
<code>/dev/appletalk</code>	directory of devices used for AppleTalk
<code>/dev/snd</code>	directory of devices providing A/UX toolbox interface to sound drivers
<code>/dev/snd/note</code>	sound driver interface to note synthesizer
<code>/dev/snd/raw</code>	raw interface for access to sound driver from CommandShell
<code>/dev/snd/reset</code>	reset “switch” for sound manager
<code>/dev/snd/samp</code>	sound driver interface to sampled synthesizer
<code>/dev/snd/wave1</code>	sound driver interface to wave synthesizer (voice 1)
<code>/dev/snd/wave2</code>	sound driver interface to wave synthesizer (voice 2)

<code>/dev/snd/wave3</code>	sound driver interface to wave synthesizer (voice 3)
<code>/dev/snd/wave4</code>	sound driver interface to wave synthesizer (voice 4)
<code>/dev/uinter0</code>	user interface device used by the A/UX toolbox
<code>/etc/RELEASE_ID</code>	A/UX Release ID number and date
<code>/etc/adduser</code>	executable shell script for <code>adduser(1M)</code> —add a user account
<code>/etc/apm_getty</code>	binary executable for <code>apm_getty(1M)</code> —set terminal type, modes, speed, and line discipline
<code>/etc/applepings</code>	binary executable for <code>applepings(1M)</code> —send test packets over the AppleTalk network
<code>/etc/appletalk</code>	binary executable for <code>appletalk(1M)</code> —configure and view AppleTalk network interfaces
<code>/etc/appletalkrc</code>	AppleTalk network configuration file (<code>appletalkrc(4)</code>)
<code>/etc/autoconfig</code>	binary executable for <code>autoconfig(1M)</code> —build a new up-to-date kernel
<code>/etc/boot.d</code>	directory of driver object modules used by <code>autoconfig(1M)</code> patched into kernel at boot time, copied from <code>/etc/install.d/boot.d</code>
<code>/etc/checkinstall</code>	executable shell script for <code>checkinstall(1)</code> —check installation of boards
<code>/etc/chgnod</code>	binary executable for <code>chgnod(1M)</code> —change current A/UX system nodename
<code>/etc/cml</code>	directory of files used by <code>autorecovery(8)</code>
<code>/etc/cml/descriptions</code>	description data base for all files as shipped (<code>cml(4)</code>)
<code>/etc/cml/init2files</code>	<code>cml(4)</code> file used by <code>autorecovery(8)</code>
<code>/etc/cml/masterlist.z</code>	data base of important info on all files as shipped (<code>cml(4)</code>)—compressed
<code>/etc/dev_kill</code>	binary executable for <code>dev_kill(1M)</code> —remove special devices from directories
<code>/etc/escher</code>	binary executable for <code>escher(1M)</code> —autorecovery administration

<code>/etc/etheraddr</code>	binary executable for <code>etheraddr(1M)</code> —get an Ethernet address
<code>/etc/ethers</code>	NIS data base. (See <code>ethers(4)</code>)
<code>/etc/eu</code>	binary executable for <code>eu(1M)</code> —update autorecovery files
<code>/etc/eupdate</code>	executable shell script for <code>eupdate(1M)</code> —update important files for autorecovery
<code>/etc/ioctl.syscon</code>	console terminal settings (<code>ioctl.syscon(4)</code>)
<code>/etc/iop</code>	directory of serial I/O Processor code resources
<code>/etc/kconfig</code>	binary executable for <code>kconfig(1M)</code> —change kernel parameters for tuning
<code>/etc/keyset</code>	binary executable for <code>keyset(1M)</code> —set console keyboard mapping
<code>/etc/line_sane</code>	binary executable for <code>line_sane(1M)</code> —push streams line disciplines
<code>/etc/loginrc</code>	executable shell script for <code>loginrc</code> (start either <code>/etc/Login</code> or <code>/etc/getty</code>)
<code>/etc/macgetty</code>	binary executable for <code>macgetty(1M)</code> —put up a Login dialog box on the console
<code>/etc/macquery</code>	binary executable for <code>macquery(1M)</code> —post a Macintosh alert to query the user
<code>/etc/macsysinit</code>	binary executable for <code>macsysinit</code> (start up the Macintosh environment during booting)
<code>/etc/macsysinitrc</code>	executable shell script for <code>macsysinitrc(1M)</code> —system initialization shell scripts
<code>/etc/newconfig</code>	executable shell script for <code>newconfig(1M)</code> —prepare and configure a new kernel
<code>/etc/newfs</code>	symbolic link to <code>/etc/fs/ufs/newfs</code> (binary executable for <code>newfs(1M)</code>)
<code>/etc/newunix</code>	executable shell script for <code>newunix(1M)</code> —prepare for new kernel configuration
<code>/etc/powerdown</code>	binary executable for <code>powerdown(1M)</code> —power down the system

<code>/etc/query</code>	binary executable for <code>query(1)</code> —query the user for input
<code>/etc/sethost</code>	set the host and domain names (shell commands called from <code>/etc/sysinitrc</code>)
<code>/etc/setport</code>	binary executable for <code>setport(1M)</code> —set a serial port
<code>/etc/settimezone</code>	binary executable for <code>settimezone(1M)</code> —set the local time zone
<code>/etc/startmsg</code>	binary executable for <code>startmsg(1M)</code> —send messages to <code>StartMonitor</code> during the A/UX boot process
<code>/etc/tty_add</code>	executable shell script for <code>tty_add(1M)</code> —modify the <code>/etc/inittab</code> file
<code>/etc/tty_kill</code>	executable shell script for <code>tty_kill(1M)</code> —modify the <code>/etc/inittab</code> file
<code>/etc/uninstall.d</code>	directory of driver removal procedures
<code>/mac/bin/Apple HD SC Setup</code>	disk formatting and partitioning application
<code>/mac/bin/CommandShell</code>	binary executable for <code>CommandShell(1)</code> —A/UX toolbox application for managing command-interpretation windows and mediating between the desktop and the A/UX environment
<code>/mac/bin/CommandShell24</code>	binary executable for <code>CommandShell24</code> (24-bit version of <code>CommandShell(1)</code>)
<code>/mac/bin/Commando</code>	Macintosh Toolbox file for <code>Commando</code> (command dialog box generator)
<code>/mac/bin/TeachText</code>	Apple <code>TeachText</code> application
<code>/mac/bin/TextEditor</code>	binary executable for <code>TextEditor(1)</code> —Macintosh-style text editor
<code>/mac/bin/changesize</code>	Macintosh Toolbox file for <code>changesize(1)</code> —change the fields of the <code>SIZE</code> resource of a file
<code>/mac/bin/cmdo</code>	binary executable for <code>cmdo(1)</code> —build commands interactively
<code>/mac/bin/derez</code>	binary executable for <code>derez(1)</code> MPW <code>derez</code> tool—decompile a resource file

<code>/mac/bin/fcnvt</code>	binary executable for <code>fcnvt(1)</code> —convert a resource file to another format
<code>/mac/bin/launch</code>	binary executable for <code>launch(1)</code> —execute a Macintosh binary application
<code>/mac/bin/mac24</code>	executable shell script for <code>mac24</code> (launch the 24-bit Macintosh environment)
<code>/mac/bin/mac32</code>	executable shell script for <code>mac32</code> (launch the 32-bit Macintosh environment)
<code>/mac/bin/rez</code>	binary executable for <code>rez(1)</code> MPW resource compiler—compile resources
<code>/mac/bin/setfile</code>	binary executable for <code>setfile(1)</code> —set file creator and type
<code>/mac/bin/startmac</code>	binary executable for <code>startmac(1)</code> —start the A/UX Finder environment
<code>/mac/bin/startmac24</code>	binary executable for <code>startmac24</code> (24-bit version of <code>startmac(1)</code>)
<code>/mac/lib</code>	directory of specialized Macintosh files
<code>/mac/lib/SystemFiles/private</code>	directory of user-writable Macintosh system files
<code>/mac/lib/SystemFiles/shared</code>	directory of shared Macintosh system files
<code>/mac/lib/cmdo</code>	directory of the Commando dialog box data base
<code>/mac/src</code>	directory of sample Macintosh application sources
<code>/mac/sys/Login System Folder</code>	directory of Macintosh initialization files related to login
<code>/mac/sys/Startup System Folder</code>	directory of Macintosh initialization files related to system startup
<code>/mac/sys/System Folder</code>	directory containing the Macintosh System file, Inits, CDEVs, and so on
<code>/newunix</code>	small copy of the kernel, used by <code>autoconfig(1M)</code>
<code>/nextunix</code>	file containing the name of the preferential kernel to boot
<code>/root</code>	alternative home directory for superuser



<code>/shlib/libmac1_s</code>	Macintosh shared libraries for code linked under A/UX 2.0.1 and later (runtime)
<code>/shlib/libmac_s</code>	Macintosh shared libraries for code linked under A/UX 2.0 (runtime)
<code>/users</code>	directory of user accounts
<code>/users/Guest</code>	the home directory for the Guest account
<code>/users/start</code>	the home directory for start, a sample account
<code>/usr/bin/at_cho_prn</code>	binary executable for <code>at_cho_prn(1)</code> —choose a default printer on the AppleTalk network
<code>/usr/bin/at_printer</code>	executable shell script for <code>at_printer(1)</code> —for backward compatibility with A/UX 1.1
<code>/usr/bin/atlookup</code>	binary executable for <code>atlookup(1)</code> —look up NVEs registered on the AppleTalk internet
<code>/usr/bin/atprint</code>	binary executable for <code>atprint(1)</code> —copy data to a remote PAP server
<code>/usr/bin/atstatus</code>	binary executable for <code>atstatus(1)</code> —get status from a PAP server
<code>/usr/bin/imagewr</code>	binary executable for <code>imagewr(1)</code> —handle international characters for ImageWriter II
<code>/usr/bin/isotomac</code>	binary executable for <code>isotomac(1)</code> —convert from Macintosh encoding to International Standards Organization (ISO) encoding
<code>/usr/bin/iw2</code>	binary executable for <code>iw2(1)</code> —Apple ImageWriter print filter
<code>/usr/bin/mactois0</code>	binary executable for <code>mactois0(1)</code> —convert from Macintosh encoding to International Standards Organization (ISO) encoding
<code>/usr/bin/module_dump</code>	binary executable for <code>module_dump(1M)</code> —dump out information from A/UX kernels
<code>/usr/bin/sumdir</code>	binary executable for <code>sumdir(1)</code> —sum and count characters in the files in the given directories
<code>/usr/bin/systemfolder</code>	executable shell script for <code>systemfolder(1)</code> —create a personal 32-bit System Folder

<code>/usr/bin/systemfolder24</code>	executable shell script for <code>systemfolder24(1)</code> —create a personal 24-bit System Folder
<code>/usr/lib/terminfo/m/mac2</code>	terminal capabilities for an Apple MacII
<code>/usr/lib/terminfo/m/mac2cs</code>	terminal capabilities for an Apple MacIIcs
<code>/usr/spool/lp/model/imagewriter2</code>	lp(1) interface for an Apple ImageWriter2
<code>/usr/spool/lp/model/iw</code>	lp(1) interface for an Apple ImageWriter
<code>/usr/spool/lp/model/psinterface</code>	lp(1) interface for Adobe PostScript®/TranScript® printer
<code>/usr/spool/lp/request/iw2</code>	ImageWriter II output spool file
<code>/usr/spool/lpd/AppleTalk</code>	directory of spooled files sent to printer “AppleTalk”
<code>/usr/spool/lpd/AppleTalk/ifilter</code>	Berkeley print spooler filter
<code>/usr/spool/lpd/AppleTalk/lock</code>	Berkeley print spooler lock file
<code>/usr/spool/lpd/AppleTalk/nroff</code>	Berkeley print spooler nroff filter
<code>/usr/spool/lpd/AppleTalk/ofilter</code>	AppleTalk printing output filter
<code>/usr/spool/lpd/AppleTalk/pipe</code>	named pipe, used for AppleTalk spooler filter
<code>/usr/spool/lpd/ImageWriter</code>	directory of spooled files sent to printer “ImageWriter”
<code>/usr/spool/lpd/ImageWriter/lock</code>	Berkeley print spooler lock file
<code>/usr/spool/lpd/LaserWriter</code>	directory of spooled files sent to printer “LaserWriter”
<code>/usr/spool/lpd/LaserWriter/lock</code>	Berkeley print spooler lock file



Appendix B: Additional Reading

This appendix provides the titles of additional books about the UNIX operating system.



- Aho, A., Kernighan, B., and Weinberger, P. *The AWK Programming Language*. Addison-Wesley, 1988
- Bach, Maurice J. *The Design of the UNIX Operating System*. Prentice Hall, 1986
- Bolsky, M. and Korn, D. *The Korn Shell Command and Programming Language*. Prentice Hall, 1989
- Bourne, S. *The UNIX System*. Addison-Wesley, 1983
- Christian, K. *The UNIX Operating System*. Wiley-Interscience
- Comer, D. *Internetworking with TCP/IP*. Prentice Hall, 1991
- Farrow. *UNIX System Security*. Addison-Wesley, 1991
- Fielder, D. and Hunter, B. *UNIX System Administration*. Hayden Books, 1986
- Garfinkel and Spafford. *Practical UNIX Security*. O'Reilly & Associates, 1991
- Hunter and Hunter. *UNIX Systems: Advanced Administration and Management Handbook*. MacMillan, 1991
- Kernighan, B. and Pike, R. *The UNIX Programming Environment*. Prentice Hall, 1984
- Kochan, S. and Wood, P. *Exploring the UNIX System*, Second edition. Hayden Books, 1989
- Kochan, S. and Wood, P. *UNIX Shell Programming*. Hayden Books, 1987
- Leffler, S., McKusick, M., Karels, M., and Quarterman, J. *The Design and Implementation of the 4.3BSD UNIX Operating System*. Addison-Wesley, 1989
- Libes, D. and Ressler, S. *Life with UNIX: A Guide for Everybody*. Prentice Hall, 1989
- Morgan, R., McGilton, H. *Introducing the UNIX System*. McGraw-Hill, 1983
- Nemeth et al. *UNIX System Administration Handbook*. Prentice Hall, 1989
- Tannenbaum, A. *Operating Systems: Design and Implementation*. Prentice Hall, 1987

Appendix C: The System V Print Spooler: `lp`

`lp` commands / C-3

Determining `lp` status / C-5

The `lp` scheduler / C-5

Configuring the `lp` system / C-8

Using the `lp` system / C-14

Troubleshooting the `lp` system / C-18

`lp` system files / C-21

This appendix covers the following topics:

- a brief explanation of the `lp` system commands
- a description of how to start and stop the `lp` scheduler
- information on managing the `lp` system through the use of the `lpadmin` command
- a brief overview of the capabilities of the `lp` system
- troubleshooting the most frequently encountered `lp` problems

The following information is provided for compatibility with the System V print spooler.

Important Before using `lp`, you must turn off the 4.3 BSD `lpr` spooler daemon (`lpd`) and clear the `lpr` print queue.

lp commands

The commands used to administer the `lp` system can be divided into two categories: those that any user can use, and those that only the `lp` administrator can use. This section gives a short description of what each command does.

Commands for general use

- `lp` Submits a print request to the `lp` system. The request is printed on the default system destination or optionally routed to a specified printer or printer class. A successful request prints a message on the user's terminal similar to
- ```
request id is dest-seqno(1 file)
```
- where *dest* is the name of a printer or printer class and *seqno* is a number unique across the entire `lp` system. See `lp(1)` in *A/UX Command Reference*.
- `cancel` Cancels requests by printer name or request ID number (*dest-seqno* supplied by `lp`). Specifying the printer name cancels the job currently printing. See `lp(1)` in *A/UX Command Reference*.
- `lpstat` Gives certain status information about the `lp` system. Also see `lpstat(1)` in *A/UX Command Reference*.
- `disable` Prevents `lpsched` from routing output requests to specified printers. See `disable(1)` in *A/UX Command Reference*.
- `enable` Allows `lpsched` to route output requests to printers. See `enable(1)` in *A/UX Command Reference*.



## Commands for lp administrators

In each lp system, at least one person must be designated as lp administrator to perform the restricted functions listed below. The lp login (provided with the standard distribution) owns all the files and commands associated with the lp system. Any user logged into the system as root or lp qualifies as an lp administrator.

The following commands are described in more detail later in this chapter:

|         |                                                                                                                                                                                                                                |
|---------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| lpadmin | Modifies the lp configuration. Many features of this command cannot be used when lpsched is running. Also see lpadmin(1M) in <i>A/UX System Administrator's Reference</i> .                                                    |
| lpsched | Routes user print requests to interface programs, which do the printing on devices. Also see lpsched(1M) in <i>A/UX System Administrator's Reference</i> .                                                                     |
| lpshut  | Stops running lpsched. All printing activity is halted, but other lp commands may still be used. Also see lpsched(1M) in <i>A/UX System Administrator's Reference</i> .                                                        |
| accept  | Allows lp to accept output requests for destinations. Also see accept(1M) in <i>A/UX System Administrator's Reference</i> .                                                                                                    |
| reject  | Prevents lp from accepting requests for particular destinations. Also see reject(1M) in <i>A/UX System Administrator's Reference</i> .                                                                                         |
| lpmove  | Moves output requests from one destination to another. Whole destinations may be moved at one time. This command cannot be used when lpsched is running. Also see lpmove(1M) in <i>A/UX System Administrator's Reference</i> . |

◆ **Note** The lp command runs with an effective user ID (EUID) of lp. In other words, it behaves as if a user named lp is reading the files to be printed. Therefore, any files to be printed with a command of the form

```
lp [options] files
```

must have read permission set for others. If this poses a security problem, you can use the lp command in a pipe (`pr filename | lp`) or with the shell input redirect character (`lp </etc/passwd`). These two methods work because the user is feeding input to the lp command via the standard input. ◆

## Determining lp status

The `lpstat` command displays on the screen the status of printing requests, destinations, and the scheduler (`lp sched`). Also see `lpstat(1)` in *A/UX Command Reference*.

Common uses of the `lpstat` command are to

- list the status of all currently printing and pending requests you have made

```
lpstat
```

- list all currently printing and pending requests of all users

```
lpstat -o
```

The status information for a request includes the request ID, the login name of the user, the total number of characters to be printed, and the date and time the request was made.

- determine whether a printer is available to print requests

```
lpstat -r -a -p
```

Before a request can be printed, the scheduler (`lp sched`) must be running and the particular printer must be enabled and accepting requests. This command produces the necessary information for all printers. Also see `accept(1M)` in *A/UX System Administrator's Reference* and `enable(1)` in *A/UX Command Reference*.

## The lp scheduler

The `lp` scheduler, `lp sched`, routes the printer requests made with `lp` through the appropriate printer interface programs and out to the printers. `lp sched` must be running and the particular printer must be enabled and accepting requests before any requests will be successfully printed. See `accept(1M)` in *A/UX System Administrator's Reference* and `enable(1)` in *A/UX Command Reference*.

## Activating the scheduler

To activate the `lp` scheduler each time the system enters multiuser mode, you need to check two files. In `/etc/rc` make sure the following line does not have a pound sign (`#`) at the start of the line:

```
rm -f /usr/spool/lp/SCHEDLOCK
```

In `/etc/inittab`, make sure that the following line appears:

```
lp:2:once:/usr/lib/lpsched >/dev/syscon 2>&1
```

## Stopping and starting the `lp` scheduler

Each time the scheduler routes a request to an interface program, it records an entry in the log file, `/usr/spool/lp/log`. This entry contains the login name of the user who made the request, the request ID, the name of the printer on which the request is being printed, and the date and time that printing first started. If a request has been restarted, more than one entry in the log file may refer to the request. The scheduler also records error messages in the log file. When `lpsched` is started, it renames `/usr/spool/lp/log` as `/usr/spool/lp/oldlog` and starts a new log file. To empty these log files of old data, use the commands

```
cp /dev/null /usr/spool/lp/oldlog
```

```
cp /dev/null /usr/spool/lp/log
```

◆ **Note** The log files can grow substantially and eventually occupy an enormous amount of disk space. You should inspect these files periodically and, if necessary, truncate them to zero length. ◆

As mentioned earlier, the `lp` system won't perform any printing unless `lpsched` is running. Use the command

```
lpstat -r
```

to find the status of the `lp` scheduler.

**Important** Do not start the `lp` scheduler while the 4.3 BSD printer spooling daemon, `lpd`, is running.

The scheduler normally begins running when the `init(1M)` process executes the entry in the `/etc/inittab` file and continues to run until the system is shut down. The scheduler operates in the `/usr/spool/lp` directory. When the scheduler starts running, it checks whether a file called `SCHEDLOCK` exists; if it does, `lpsched` exits immediately. Otherwise, `lpsched` creates `SCHEDLOCK` to prevent more than one scheduler from running at the same time.

Occasionally, it is necessary to shut down the scheduler to reconfigure the `lp` software. To stop the scheduler, give the command

```
/usr/lib/lpshut
```

This stops `lpsched`, removes the `SCHEDLOCK` file, and terminates all printing. All requests that were in the middle of printing will be reprinted in their entirety when you restart the scheduler.

To restart the `lp` scheduler, use the command

```
/usr/lib/lpsched
```

Shortly after you enter this command, you can use the `lpstat -r` command to determine whether the scheduler is running. If not, it is possible that A/UX crashed or was halted improperly, leaving `SCHEDLOCK` in the `/usr/spool/lp` directory.

Use the command

```
ls /usr/spool/lp/SCHEDLOCK
```

to determine whether `SCHEDLOCK` exists. If it does, enter the commands

```
rm -f /usr/spool/lp/SCHEDLOCK
```

```
/usr/lib/lpsched
```

Wait about 15 seconds and then use the `lpstat -r` command to determine if the scheduler is running.

# Configuring the `lp` system

The `lp` system configuration is determined by a set of data files in the `/usr/spool/lp` directory. Some of these files are ordinary text files, and others contain binary data.

- ⚠ **Important** Although it is possible to change the text files with a text editor, you should not do so. Altering these files by hand while `lpsched` is running may cause strange spooler behavior. The `lpadmin` command is designed with these conditions in mind and will not allow certain configuration changes to take place until you terminate `lpsched`. Always use the `lpadmin` command to reconfigure the `lp` system.

The `lpadmin` command is used to change the content of these files. The `lpadmin` command can have one of the following forms:

```
lpadmin -pprinter [-vdevice] [options]
```

```
lpadmin -xdest
```

```
lpadmin -d[dest]
```

The three flag options `-p`, `-x`, and `-d` are mutually exclusive. Also, `lpadmin` will not attempt to alter the `lp` configuration when `lpsched` is running, except as explicitly noted in the rest of this section.

The rest of the section is devoted to a series of examples that illustrate possible invocations of the commands in the `lp` system.

## Introducing new destinations

You can add a new printer by giving the command

```
lpadmin -pprinter [-vdevice] [options]
```

where the fields are interpreted as follows:

- printer* Arbitrary name that must
- contain no more than 14 characters
  - contain only alphanumeric characters and underscores
  - not be the name of an existing `lp` destination (whether a printer or a class)
- device* A **hard-wired** printer (plugged directly into the computer) or other file that can be written to by `lp`; for example, `/dev/printer`.
- options* Any of the following:
- `-c class` Inserts the specified *printer* into the class *class*. The class will be created if it does not already exist.
  - `-e printer-to-copy` Copies the interface program for *printer-to-copy* as the new interface program for *printer*.
  - `-h` Indicates that the device associated with *printer* is hard-wired. This option is always assumed, unless the `-l` option is selected.
  - `-i interface` Establishes the program found in *interface* as the new interface program for *printer*.
  - `-l` Indicates that the device associated with *printer* is a login terminal.
  - `-m model` Selects *model* as the model interface program for *printer*.
  - `-r class` Removes printer *printer* from the specified class. If the specified printer is the last member of the class, the class will also be removed.
  - `-v device` Associates a new device *device* with the *printer*. *device* must be a pathname of a file that can be written to by `lp`.

When adding a new printer to the `lp` system, you must select the printer interface program. You may specify this in one of three ways:

- You may select it from a list of model interfaces supplied with `lp` in the `/usr/spool/lp/model` directory (`-m model`).
- It may be the same interface that an existing printer uses (`-e printer-to-copy`).
- It may be a program supplied by the `lp` administrator (`-i interface`).

You may add the new printer to an existing class or to a new class (`-c class`). New class names must conform to the rules that govern new printer names.

Here are some examples of how printers might be named:

**Create a printer called `pr1` whose device is `/dev/printer` and whose interface program is the model `hp` interface:**

```
/usr/lib/lpadmin -ppr1 -v/dev/printer -mhp
```

**Add a printer called `pr2` whose device is `/dev/tty1` and whose interface is a variation of the model `prx` interface:**

```
cd /usr/spool/lp/model/prx
```

```
cp prx newint
```

Edit `newint` and introduce modifications

```
/usr/lib/lpadmin -ppr2 -v/dev/tty1 -inewint
```

**Create a printer called `pr3` whose device is `/dev/tty1`. Printer `pr3` will be added to a new class called `c11` and will use the same interface as printer `pr2`:**

```
/usr/lib/lpadmin -ppr3 -v/dev/tty1 -epr2 -cc11
```

## Modifying existing destinations

You can use `lpadmin` to modify existing destinations. Always make modifications with respect to a printer name (`-pprinter`). The form of the command is

```
lpadmin -pprinter options
```

These are the options available for modifying existing destinations:

- c *class*            Adds the printer to a new or existing class.
- e *printer-to-copy*   Changes the printer interface program to the one used by *printer-to-copy*.
- i *interface*        Changes the printer interface program to the full pathname of the file specified by *interface*.
- m *model*            Changes the printer interface program to the file in the *model* directory.
- r *class*            Removes the printer from an existing class. Removing the last remaining member of a class causes the class to be deleted. A destination cannot be removed if there are pending requests to that destination. In that case, you should use `lpmove` or `cancel` to move or delete the pending requests.
- v *device*          Changes the device for the printer. If this is the only modification, this may be done even while `lpsched` is running.

The following examples are based on the `lp` configurations created in previous examples:

**Add printer `pr2` to class `cl1`:**

```
/usr/lib/lpadmin -ppr2 -cc11
```

**Change the interface program of the `pr2` to the model `prx` interface, change its device to `/dev/tty0`, and add it to a new class called `cl2`:**

```
/usr/lib/lpadmin -ppr2 -mprx -v/dev/tty0 -cc12
```

Printers `pr2` and `pr3` now use different interface programs, even though `pr3` was originally created with the same interface as `pr2`. Printer `pr2` is now a member of two classes.



**Add printer `pr1` to class `c12`:**

```
/usr/lib/lpadmin -ppr1 -cc12
```

The members of class `c12` are now `pr2` and `pr1`, in that order. Requests routed to class `c12` will be serviced by `pr2` if both `pr2` and `pr1` are ready to print; otherwise, they will be printed by whichever one is next ready to print.

**Remove printers `pr2` and `pr3` from class `c11`:**

```
/usr/lib/lpadmin -ppr2 -rc11
```

```
/usr/lib/lpadmin -ppr3 -rc11
```

Because `pr3` was the last remaining member of class `c11`, the class is removed.

**Add `pr3` to a new class called `c13`:**

```
/usr/lib/lpadmin -ppr3 -cc13
```

## Altering the system default destination

You can change or specify the system default destination even when `lpsched` is running. You can do this using the `lpadmin` command with the `-d` option. The form of the command is

```
lpadmin -d[dest]
```

The destination *dest* may be omitted; if so, then no destination is established as the system default.

Here are some examples of how default destinations may be specified:

**Establish class `c11` as the system default destination:**

```
/usr/lib/lpadmin -dc11
```

**Establish no default destination:**

```
/usr/lib/lpadmin -d
```

## Removing destinations

You can use `lpadmin` to remove classes and printers only if there are no pending requests routed to them. You must either use `cancel` to cancel pending requests or use `lpmove` to move pending requests to other destinations before you can remove destinations. If the removed destination is the system default destination, the system has no default destination until you specify a new default destination. When the last remaining member of a class is removed, the class is also removed. In contrast, removing a class never implies removing printers (see the third example, following).

The form of the `lpadmin` command used to remove destinations is

```
lpadmin -xdest
```

The destination being removed must be specified, and no other options are allowed.

Here are some examples of how printer destinations may be removed:

### **Make printer `pr1` the system default destination:**

```
/usr/lib/lpadmin -dpr1
```

Then remove printer `pr1`:

```
/usr/lib/lpadmin -xpr1
```

Now there is no system default destination.

### **Remove printer `pr2`:**

```
/usr/lib/lpadmin -xpr2
```

Class `c12` is also removed because `pr2` was its only member.

### **Remove class `c13`:**

```
/usr/lib/lpadmin -xc13
```

Class `c13` is removed, but printer `pr3` remains.

# Using the `lp` system

Once `lp` destinations have been created, users may route output to a destination by using the `lp` command. The basic form of the `lp` command is

```
lp [options] files
```

Various options are available with the `lp` command; see `lp(1)` in *A/UX Command Reference*. You can use the request ID returned by `lp` to see if the request has been printed or to cancel the request.

The `lp` program determines the destination of a request by checking the following list in order:

- If the user specifies `-ddest` on the command line, the request is routed to *dest*.
- If the environment variable `LPDEST` is set, the request is routed to the value of `LPDEST`.
- If there is a system default destination, the request is routed there.
- Otherwise, the request is rejected.

Here are some examples of using the `lp` command.

- There are at least four ways to print a file on the system default destination:

```
lp filename
```

```
lp < filename
```

```
cat filename | lp
```

```
lp -c filename
```

In the first method, the file is printed directly; in the last three, the file is printed indirectly. If you use the first command and the file is modified between the time the request is made and the time it is actually printed, the changes will be reflected in the output.

- Invoke the `pr` command on the file `abc` and pipe the output to `lp`, which prints two copies on printer `iw2` and calls the output `myfile`:

```
pr abc | lp -diw2 -n2 -t "myfile"
```

- Print file `abc` on a printer called `jerry` in 12 pitch, and write the file to the user's terminal when printing is completed:

```
lp -djerry -o12 -w abc
```

In this example, `l2` is a command to the printer interface program to print output in 12-pitch mode. Various printer models may require different options. See `lpadmin(1M)` in *A/UX System Administrator's Reference*.

## Allowing and refusing requests

When a new destination is created, `lp` at first rejects requests that are routed to it. When you are sure that the new destination is set up correctly, you should use the `accept` command to allow `lp` to accept requests for that destination. See `accept(1M)` in *A/UX System Administrator's Reference*.

Sometimes it is necessary to prevent `lp` from routing requests to destinations. If printers have been removed or are waiting to be repaired, or if too many requests are in queue for printers, you may want to have `lp` reject requests for those destinations. The `reject` command performs this function; see `reject(1M)` in *A/UX System Administrator's Reference*. After the condition has been remedied, you should use the `accept` command to allow requests to be taken again.

The acceptance status of destinations is reported by the `-a` option of `lpstat`. Here are some examples of how to reject and accept requests:

### **Have `lp` reject requests for destination `iw2`:**

```
/usr/lib/reject -r "printer iw2 needs repair" iw2
```

Users who try to route requests to `iw2` see the following message:

```
lp -iw2 file
lp: cannot accept requests for destination "iw2"
 -- printer iw2 needs repair
```

### **Allow `lp` to accept requests routed to destination `iw2`:**

```
/usr/lib/accept iw2
```

## Allowing and inhibiting printing

The `enable` command allows the `lp` scheduler to print requests on printers. The scheduler routes requests only to the interface programs of enabled printers. By issuing the appropriate `enable` and `reject` commands, you can enable a printer and at the same time prevent further requests from being routed to it. This can be useful for testing purposes.

The `disable` command reverses the effects of the `enable` command. It prevents the scheduler from routing requests to printers, regardless of whether `lp` is allowing them to accept requests. Printers may be disabled for several reasons, including malfunctioning hardware, paper jams, and end-of-day shutdown. If a printer is printing a request at the time it is disabled, the request will be reprinted in its entirety either on another printer (if the request was originally routed to a class of printers) or on the same one when the printer is enabled once again.

The `-c` option cancels the currently printing requests on busy printers in addition to disabling the printers. This is useful if strange output is causing a printer to behave abnormally.

Here are some examples of how to enable and disable a printer:

### **Disable printer `iw2` because of a paper jam:**

```
disable -r "paper jam" iw2
```

The `disable` command prints this status message:

```
printer "iw2" now disabled
```

### **Find the status of printer `iw2`:**

```
lpstat -piw2
```

The `lpstat` command prints this status message:

```
printer "iw2" disabled since Jan 5 10:15 -
 paper jam
```

### **Reenable iw2:**

```
enable iw2
```

The `enable` command prints this status message:

```
printer "iw2" now enabled
```

## Moving requests between destinations

Occasionally, `lp` administrators find it useful to move output requests between destinations. For instance, when a printer is down for repairs, you may want to move all of its pending requests to a working printer. This is one use of the `lpmove` command. The other use of this command is moving specific requests to a different destination. The `lpmove` command will not move requests while the `lp` scheduler is running.

Here are some examples of how to move requests between destinations:

### **Move all requests for printer abc to printer bobby:**

```
/usr/lib/lpshut
/usr/lib/lpmove abc bobby
/usr/lib/lpsched
```

The names of all the moved requests are changed from `abc-mm` to `bobby-mm`. As a side effect, destination `abc` will not accept further requests.

### **Move requests jerry-543 and abc-1200 to printer bobby:**

```
/usr/lib/lpshut
/usr/lib/lpmove jerry-543 abc-1200 bobby
/usr/lib/lpsched
```

The two requests are now renamed `bobby-543` and `bobby-1200` and will be printed on `bobby`.

## Canceling requests

To cancel `lp` requests, use the `cancel` command. The `cancel` command can take two types of arguments: request IDs and printer names. Requests identified by request IDs are canceled. If you use a printer name as the argument to `cancel`, all jobs currently printing on the named printers are canceled. The two arguments may be intermixed. See `lp(1)` in *A/UX Command Reference*.

Here is an example of how to cancel printing:

**Cancel the request that is now printing on printer `bobby`:**

```
cancel bobby
```

If the user who cancels a request is not the user who made it, mail is sent to the owner of the request. The `lp` scheduler allows any user to cancel requests, eliminating the need for the user to find an `lp` administrator when unusual output should be stopped.

## Troubleshooting the `lp` system

The `lp` system problems encountered most frequently are explained here, along with their solutions.

### Problems starting `lpsched`

The `lpsched` scheduler is usually invoked by the `init(1M)` process when A/UX enters multi-user mode. The invocation is a two-step process:

- The `/etc/rc` script runs `rm` to remove the `SCHEDLOCK` file in the `/usr/spool/lp` directory.
- The `init` process invokes `lpsched`.

The purpose of the `SCHEDLOCK` file is to prevent more than one invocation of `lpsched` from running simultaneously. If two or more copies of `lpsched` are running at the same time, there is contention over system resources, resulting in confused spooler behavior and failure to print files.

When `lpsched` finds something wrong in the `lp` system, it attempts to mail an error message to `root` and to make an entry in the `/usr/spool/lp/log` file. The `SCHEDLOCK` file is not removed under these conditions, because invoking `lpsched` again without clearing the trouble is likely to produce the same error conditions.

## Restarting `lpsched`

### 1 **When `lpsched` stops due to error conditions:**

- Check the mail for `root` to see if it contains correspondence from `lp`.
- Check the `/usr/spool/lp/log` for error messages.
- Use `lpstat -t` to check the spooler status for additional messages about individual printers.
- Use the `ps -u lp` command to determine if multiple copies of `lpsched` are running. (The status command `lpstat` will not report multiple copies of `lpsched`.) Write down the process ID of each `lpsched` you find.
- Use the `kill(1)` command to kill all of the `lpsched` processes. (See `kill(1)` in *A/UX Command Reference*.)

### 2 **Clear the error conditions:**

- If messages from `lpsched` indicate damaged spooler configuration files (see “`lp` System Files,” later in this appendix), use the `lpadmin` command to remake the `lp` system (see “Configuring the `lp` System,” earlier in this appendix).
- Clear any other error conditions indicated by the error messages.



### 3 **Restart** `lpsched` **with the commands**

```
rm /usr/spool/lp/SCHEDLOCK

/usr/lib/lpsched
```

Use the `lpstat -t` command to check the status of the entire `lp` system.

### 4 **If everything appears normal in the** `lpstat` **report, perform the ultimate test: print a file.**

## Repairing a damaged `outputq` file

The `lp` system keeps all queue data in the binary file `/usr/spool/lp/outputq`. If this file is damaged, the spooler does not run correctly, and old job files remain in the subdirectories of `/usr/spool/lp/request`. To correct this condition:

### 1 **Use the** `fsck` **utility to check the** `/usr` **file system.**

See Chapter 8, “Checking the A/UX File System: `fsck`.”

### 2 **Use the** `/usr/lib/lpshut` **command to stop the** `lp` **spooler.**

### 3 **Remove the contents of the directories under** `/usr/spool/lp/request`.

◆ **Note** Do not remove the directories themselves. Use the `rm ./*` command with caution. Be sure you are in the directory you think you are in by checking with the `pwd` command. ◆

### 4 **Nullify the corrupted** `outputq` **file with the command**

```
cp /dev/null /usr/spool/lp/outputq
```

### 5 **Use the** `/usr/lib/lpsched` **command to restart the spooler.**

## lp system files

This section describes the system files used by `lp`.

`/usr/spool/lp/class`

A directory containing one text file for each printer class. The filename corresponds to the class. Each class file contains the names of the printers belonging to the class.

`/usr/spool/lp/default`

A text file containing the name of the system default printer, empty if there is no default printer or destination.

`/usr/spool/lp/log`

A text file containing a record of all printing requests.

`/usr/spool/lp/FIFO`

A named pipe that can be read from and written to only by `lp`. Any `lp` command can write to this file, but only `lpsched` can read it.

`/usr/spool/lp/interface/printer`

The *printer* field is the name of a particular printer interface program in the `/usr/spool/lp/interface` directory. All files in this directory should be executable by `lp` only.

`/usr/spool/lp/log`, `/usr/spool/lp/oldlog`

The file `/usr/spool/lp/log` is a record of printing requests made during each run of `lpsched`. Each time `lpsched` is started, it copies `/usr/spool/lp/log` to `/usr/spool/lp/oldlog`. Then it truncates `/usr/spool/lp/log`.

`/usr/spool/lp/member`

A directory containing one text file for each printer. The file name corresponds to the printer name. The file contains the name of the device file in the `/dev` directory that corresponds to the printer.

`usr/spool/lp/model`

A directory containing sample printer interface programs (Bourne shell scripts).

`/usr/spool/lp/outputq`

A binary data file that holds the `lp` request queue information.

`/usr/spool/lp/pstatus`

A binary data file that contains status information (whether a printer is enabled or disabled) for each printer.

`/usr/spool/lp/qstatus`

A binary data file that contains the acceptance status (whether a printer is accepting or rejecting requests) for each printer.

`/usr/spool/lp/request`

A directory containing subdirectories named for each destination (class or printer) known to the `lp` system. The subdirectories are used for temporary storage of spooler commands and print requests (text).

`/usr/spool/lp/SCHEDLOCK`

A file designed to prevent more than one invocation of `lpsched` from running simultaneously. See “Stopping and Starting the `lp` Scheduler,” earlier in this appendix.

`/usr/spool/lp/seqfile`

A text file containing the sequence number assigned to the last request printed. The number is always in the range 1–9999.

`/usr/spool/lp/OUTQLOCK`

`/usr/spool/lp/PSTATLOCK`

`/usr/spool/lp/QSTATLOCK`

`/usr/spool/lp/SEQLOCK`

Various lock files for preventing `lp` system commands from modifying data in the data files described above. Each file has an expiration time, after which any `lp` system command may remove the `lock` file and then modify the previously locked data file. These lock files and `SCHEDLOCK` operate according to similar principles.

## lp system command permissions

All `lp` system utilities except for `lpsched` should be owned by `lp` with the `setuid` bit turned on (see `chmod(1)` and `chown(1)` in *UNIX Command Reference*). The `lpsched` scheduler may be owned by either `root` or `lp`.



# Appendix D: System Accounting Package

Overview / D-3

Starting system accounting / D-3

Accounting reports / D-6

Routine accounting procedures / D-15

Stopping system accounting / D-25

The system accounting package collects detailed information on system usage and allows you to generate various system usage reports on a periodic basis. These reports can reveal usage patterns both of the overall system and of individual users. This reporting function can be automated; the procedure is described in the section “Starting System Accounting.” You can also produce customized reports; these are described in the section “Accounting Reports,” later in this chapter.

# Overview

The system accounting package was developed on large UNIX systems where there were usually several people logged in at once and there was a need to allocate the costs associated with the computer. Generally on A/UX systems there is only one user logged in at a time, and the cost of computing has decreased by several orders of magnitude. Thus, the original reason for an accounting function has all but disappeared. However, system accounting can still be used to track the costs associated with individual projects.

Using the system accounting package you can track the amount of CPU time, connect time, memory usage, disk usage, and any special fees. Provision has been made to track usage during both prime time (usually the hours of peak computer load or the hours of the normal working day) and nonprime time, in case you want to charge users different fees for using the system during those periods.

## Starting system accounting

The system is shipped with accounting turned off. You need only modify a few files to start the accounting function. Take the following steps to turn on system accounting and automate its operation:

- 1 **Log in as root.**
- 2 **Remove the leading # from the following lines in the `/etc/rc` file:**

```
/bin/su adm -c /usr/lib/acct/startup
echo process accounting started
```

The first line is a command that activates the accounting system when the system is brought up. The second line prints `process accounting started` when the accounting system starts up.

- 3 **Change your effective user name to *adm*.**

Enter the command:

```
su adm
```

#### 4 Copy the current crontab file.

Enter the command:

```
crontab -l > /tmp/adm.cron
```

to put a copy of the current crontab file into the file `/tmp/adm.cron`.

#### 5 Edit the file `/tmp/adm.cron` to reflect the new changes. Remove the leading `#` from the following lines:

```
0 4 * * 1-6 /usr/lib/acct/runacct 2> /usr/adm/acct/nite/fd2log
0 2 * * 4 /usr/lib/acct/dodisk
5 * * * * /usr/lib/acct/ckpacct
15 5 1 * * /usr/lib/acct/monacct
```

The instructions direct `cron` to run the daily accounting automatically. See the section “The Crontab File Format,” in Chapter 5, for a description of this file.

#### 6 Notify the operating system to read the modified file.

Enter the command:

```
crontab /tmp/adm.cron
```

This copies the file into `/usr/spool/cron/crontabs/adm` and alerts `cron` to reread the crontab file. The edited entries cause the accounting functions to operate. It is the responsibility of the system administrator to check the desired reporting function regularly and to copy reports for later review.

#### 7 Verify that the following line is in the `/usr/adm/.profile` file:

```
PATH=/usr/lib/acct:/bin:/usr/bin
```

This ensures that the operating system has access to all the necessary files and scripts to process the accounting system automatically.

#### 8 Update the `/usr/lib/acct/holidays` file.

This file defines the prime-time for your system and the holidays recognized by the system accounting function. See the section “Updating Holidays,” later in this chapter, for a description of this file.

The system will start the accounting function after the next restart.



## Updating holidays

The file `/usr/lib/acct/holidays` contains the prime/nonprime table, which gives the start of prime time and the start of nonprime time for your operating system, using a 24-hour system (0100 to 2400 hours). Information on changing the prime/nonprime table to reflect individual preferences is given in this section. For example, during normal business operation, prime time is considered to be from 8:30 A.M. to 5:30 P.M. The table also contains the holiday schedule for the year, which you can adjust to include additional holidays.

The format of the `holidays` file includes

- comment lines

Comment lines can appear anywhere in the file, but they must be preceded by an asterisk so that the system won't read them.

- year designation line

The year designation line must be the first data line in the file and can appear only once in the file. It consists of three tab-separated fields of four digits each:

```
yyyymmhh mmhhmm
```

The first field is the current year. Be sure to set it correctly, because an incorrect year entry affects the holiday entries.

The second field is the start of prime time, in 24-hour time. Prime time refers to the peak activity period for your system. In most businesses, for example, prime time starts at 8:30 A.M. (0830 in a 24-hour system) to coincide with the start of the business day.

The third field is the start of nonprime time, in 24-hour time. This field represents the end of peak activity for the operating system, usually 5:30 P.M. (or 1730).

For example, you can set your system to start prime time at 0800 and nonprime time at 1700 (5:00 P.M.). Or, if your company begins work at 8:00 P.M., you may want to change the start of prime time to 2000. You would also change the nonprime time to reflect the close of business, say 2:00 A.M., as 0200.

- ◆ **Note** The hour 2400 automatically converts to 0000. ◆

- company holiday lines

You can make entries for national and local holidays on the line following the year designation line. The format is

*day-of-year month day description-of-holiday*

The *day-of-year* is a number from 1 to 366, indicating the day for the corresponding holiday. The other three fields are not used by A/UX and are provided for commentary only.

- ◆ **Note** Remember to separate all entries with tabs and not with spaces. ◆

## Accounting reports

The system provides for several reports, some of which are produced automatically by the commands present in the `/usr/spool/cron/crontabs/adm` file.

- The `prdaily` program prints a report of the previous day's accounting.
- The `acctcom` program prints detailed information on processes run by a particular user, or group, or in a specific time period.
- The `prtacct` program formats and prints any total accounting file.
- The `prctmp` program prints the login session record file.

### Daily summary reports

The script `prdaily` is run automatically, usually every day (as defined in the crontab file), and generates a report based on the `runacct` process. The report resides in `/usr/adm/acct/sum/rprt` *mmdd*. The command syntax is

```
prdaily [-l] [-c] [mmdd]
```

The notation *mddd* indicates the month and day of the report. You can generate previous daily reports using this option, specifying the month and day of the data you wish to see. Note that the report generated on 0611, for example, is actually the report on usage for 0610. Daily information is no longer available after `monacct`, which produced a monthly summary report, is run.

The `-l` flag prints a report of exceptional usage by login name for a date specified with *mddd*. The definition of “exceptional” usage by login name is provided in the file `/usr/lib/acct/ptelus.awk`. These values are considered to signal exceptional usage: `CPU > 20`, `KCORE > 500`, and `CONNECT > 120`.

The `-c` flag prints a report of exceptional resource usage by command. You can use it on the current day’s accounting data only. The definition of exceptional usage by command is provided in the file `/usr/lib/acct/ptecms.awk`.

To read the daily report, enter

```
/usr/lib/acct/prdaily | more
```

The preceding command displays a report generated by the `runacct` procedure. The report consists of a header and five parts.

The header displays the dates of the current reporting period; for example,

```
Apr 01 10:04 1992 DAILY REPORT FOR A/UX Page 1
from Tue Mar 31 04:00:13 1992
to Wed Apr 01 04:00:13 1992
```

Following the date is a listing of the `/etc/wtmp` entries generated by the `acctwtmp` program. This listing includes any reboots, shutdowns, power failure recoveries, date changes, and so on that occurred during the reporting period, for example,

```
2 date changes
```

Commands used to produce the report are displayed; for example,

```
1 runacct
1 acctconl
```

The first part of the report describes the connect accounting information on terminal usage: the number of sessions (logins, logouts) and the amount of time each terminal was used. (CommandShell windows are considered separate terminals.) The fields in this part of the report are

|                |                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
|----------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| TOTAL DURATION | The amount of time the system was in multi-user mode.                                                                                                                                                                                                                                                                                                                                                                                           |
| LINE           | The terminal line or access port used.                                                                                                                                                                                                                                                                                                                                                                                                          |
| MINUTES        | The amount of time the line was in use during the reporting period.                                                                                                                                                                                                                                                                                                                                                                             |
| PERCENT        | The value of MINUTES divided by TOTAL DURATION.                                                                                                                                                                                                                                                                                                                                                                                                 |
| # SESS, # ON   | These columns give the number of logins on the line during the reporting period. This report is helpful in finding which lines have been logged on but not off.                                                                                                                                                                                                                                                                                 |
| # OFF          | Not only the number of logoffs but also the number of interrupts on the line. If the # OFF exceeds the # ON by a large factor, it is possible that there is a bad connection or that a multiplexer, modem, or cable is going bad. Monitor the <code>/etc/wtmp</code> file; if it grows rapidly, execute <code>acctcon1</code> to see which <code>tty</code> line is the noisiest. A large number of interrupts can affect the system adversely. |

The next part of the report is a breakdown of system resource use by user. It does not give specific information on what each user was doing but provides information on the CPU time and connect time for each user. To charge users for system usage, see the information in the `FEE` column and the “The `chargefee` Procedure,” later in this chapter.

|            |                                                                                                                                                                                                                             |
|------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| UID        | The user ID (UID) for each login. For more information on UID, see Chapter 3, “User and Group Administration.”                                                                                                              |
| LOGIN NAME | The actual user login name. This column lets you differentiate the activities of users with the same UID. The next column gives CPU usage. This figure is broken down into two amounts: prime time and nonprime time usage. |
| KCORE-MINS | A cumulative measure of the amount of memory a process uses. It is measured in kilobytes per minute and is broken down into prime time and nonprime time usage.                                                             |

|                |                                                                                                                                                                                                                                                                                                       |
|----------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| CONNECT (MINS) | The amount of time a user was logged in. It is measured in “real time” and is broken down into prime time and nonprime time usage. If this number is high, and the number in the # OF PROCS column is low, the user probably logs in first thing in the morning and then hardly uses the terminal.    |
| DISK BLOCKS    | An accounting of the disk usage by user as it is calculated by the <code>acctdusg</code> program. The # OF PROCS column gives the number of processes used. A very high number might indicate a shell process in an infinite loop. The # OF SESS column tells you how many times each user logged in. |
| DISK SAMPLES   | The number of times the <code>acctdusg</code> ran to obtain the information in the DISK BLOCKS column.                                                                                                                                                                                                |
| FEE            | A record of special charges to each user ID. This information is written to <code>/usr/adm/fee</code> and merged with the other accounting records during the night. It then appears in the FEE column in the daily report.                                                                           |

The next two parts of the report, the `DAILY COMMAND SUMMARY` and the `MONTHLY COMMAND SUMMARY`, summarize command usage over the report period. The report period is specified in the *number* argument to the `monacct` command; if it is not specified, the default is the current month. These parts are identical in format. The daily report summarizes the command usage for the report period, and the monthly report summarizes the command usage from the beginning of the month through the current period. Entries may appear in the monthly command name column that do not appear in the daily report. These are commands that were used some time during the current month but not during the current reporting period.

The columns and their output are defined as follows:

#### DAILY, MONTHLY, and TOTAL KCOREMIN

Both the `DAILY` and `MONTHLY` command summaries are sorted by the `TOTAL KCOREMIN` column, which provides the total amount of memory a process uses per minute, measured in kilobytes. You can change the sort order by modifying the `runacct` script.

COMMAND NAME

Self-explanatory. All shell commands, however, are lumped under the entry `sh`. For example, the command `cd` won't appear in the report; it is included in the `sh` column. All purely Macintosh applications are lumped together under the entry `startmac`. This means that the system accounting function cannot tell you the usage of each Macintosh application.

◆ **Note** Entries such as `a.out`, `core`, or mysterious command names indicate errors in compiled programs. If a compiled program is not named, it appears in the report under the default name `a.out`. The name `core` indicates errors in the execution of a compiled program. You can use `acctcom` to tell you who used a suspicious command, perhaps an alias, or who has been exercising root user account privileges. See the section "Detailed Reports," later in this chapter, for more information. ◆

NUMBER CMNDS

Total number of times a command was executed.

TOTAL CPU-MIN

Total processing time dedicated to a command.

TOTAL REAL-MIN

Time it takes to process the command in real time, including the real-time usage of background processes.

MEAN SIZE-K

Calculated as  $\text{TOTAL KCOREMIN} / \text{NUMBER CMNDS}$ .

MEAN CPU-MIN

Calculated as  $\text{NUMBER CMNDS} / \text{TOTAL CPU-MIN}$  used to execute the commands.

#### HOG FACTOR

The ratio of system availability to system utilization. Calculated by dividing the total CPU time by the elapsed time, it provides a relative measure of the CPU time the process used during its execution.

#### CHARS TRNSFD

The total number of characters transferred by the read and write system calls. The number of characters is calculated command by command. It can be a negative number; for example, the reads may outnumber the writes.

#### BLOCKS READ

A total count of the physical block reads and writes that a process performs.

The last part of the accounting report is a compilation of all the logins on the system and the last date they logged in. The report looks like this:

```
Apr 01 04:05 1992 LAST LOGIN Page 1
92-03-25 apple
92-03-27 alice
00-00-00 phil
00-00-00 sys
92-03-29 john
```

The first column gives the date of the last login in *yy-mm-dd* format. The second column gives the login name itself. As you can see from the example, the date information can be blank. If the system is shut down for any reason, the last login date for all users who have not logged on since the shutdown is `00-00-00`. You can use this part of the report to determine which users are no longer active. These users (excluding those who may have logged on prior to a crash but not since) may be candidates for removal.

Of course, if your system date is set incorrectly, or the battery dies, all the information is potentially wrong.

## Detailed reports

Besides the summary user information you can get from the automated procedures, additional information about users is available. For example, the summary report does not tell you who is doing what, or when. One way to gather this information is by implementing additional accounting commands supplied with your system.

The `acct.com` command is probably the most useful accounting command supplied with your system. It reports what processes are associated with a particular terminal, user, or group of users.

The command syntax of `acct.com` is

```
acct.com [options] [file. . .]
```

The `acct.com` command reads a specified file, the standard input, or `/usr/adm/pacct.`, and writes the selected records to the standard output. Each record represents the execution of one process.

If you don't specify a file, and standard input is associated with a terminal, `/usr/adm/pacct` is read. If this isn't the case, the standard input is read. If *file* arguments are given, they are read in the order given. Each individual file is read in chronological order by process completion time. The `/usr/adm/pacct` file is usually the current file to be examined. A busy system, however, may have several `pacct` files to be processed.

The output generated by this command is similar in format to the report generated by `runacct`. It includes the following column headings:

|              |                                                                                                                                                                                                                                                                                       |
|--------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| COMMAND NAME | The command name is preceded by a number sign (#) if the command was executed with <code>root</code> user account privileges. By using the <code>-u</code> option of the command, you can find out which users (selected with the <code>-u</code> option) are executing the commands. |
| USER NAME    | The login name of the user.                                                                                                                                                                                                                                                           |
| TTY NAME     | The terminal associated with the process. If a process is not associated with a known terminal, a period (.) appears in this column.                                                                                                                                                  |
| START TIME   | The time the process began.                                                                                                                                                                                                                                                           |
| END TIME     | The time the process terminated.                                                                                                                                                                                                                                                      |



|               |                                                                                                              |
|---------------|--------------------------------------------------------------------------------------------------------------|
| REAL (SEC)    | The elapsed real time the process took to complete.                                                          |
| CPU (SEC)     | The elapsed CPU time the process took to complete.                                                           |
| MEAN SIZE (K) | The average amount of memory (in kilobytes) used by a process over the number of invocations of the process. |

The following information appears in the output if certain options are used with `runacct`. The options are explained next.

|              |                                                                                                           |
|--------------|-----------------------------------------------------------------------------------------------------------|
| STAT         | The system exit status.                                                                                   |
| HOG FACTOR   | Ratio of system availability to system utilization. It is calculated as total CPU time over elapsed time. |
| KCORE MIN    | The amount of kilobyte segments of memory used by a process.                                              |
| CPU FACTOR   | A measurement of user time over system time plus user time.                                               |
| CHARS TRNSFD | The number of characters transferred by the <code>read</code> and <code>write</code> commands.            |
| BLOCKS READ  | A total count of the physical block reads and writes that a process performed.                            |

The `acctcom` command can be used with several options. Here is a partial list of options with a brief explanation of what each does. Try a few to find out which ones are best suited to your purposes. Pay particular attention to the `-u` option, which describes user usage of the system. For a full listing of all the options, see `acctcom(1M)` in *A/UX System Administrator's Reference*.

|    |                                                                                                                                                                                                   |
|----|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| -a | The main <code>acctcom</code> option. In addition to printing the column headings in the preceding list, it prints some average statistics about the processes selected at the end of the report. |
| -f | Prints a report with columns showing the number of <code>fork/exec</code> flags and the system exit status.                                                                                       |
| -h | Displays the fraction of total available CPU time consumed by the process during its execution in a column headed <code>HOG FACTOR</code> .                                                       |
| -l | Since the console is used almost exclusively on A/UX systems, this option is not extremely useful. If you specify this option, you always get information for all terminals.                      |

- u Gives you a report of all system usage by a particular login name. The option requires the argument *user*, which specifies the login name about which you wish to generate a report. You can use it in conjunction with the *-s*, *-e*, *-S*, and *-E* options to limit the search to a specific time period. If you specify an incorrect login name, *-u* generates an error message and then produces the entire report anyway.
- g Similar to the *-u* option. Instead of printing system usage by user, however, it prints usage by the group. It requires the argument *group*, which may be either the group name or group ID. The */etc/group* file, of course, must be correct for this option to work properly.
- s, -S, -e, -E Limits the reported information to processes that occur by a specified time. These options can be used with the other options to specify a range of time to which the report of activities will apply. These options require the argument  
*hr[:min[:sec]]*
  - s selects processes existing at or after the time.
  - S selects processes starting at or after the time.
  - e selects processes existing at or before the time.
  - E selects processes ending at or before the time.

## The `prtacct` procedure

The script `prtacct` prints total accounting files, that is, files formatted in the `taacct` format. The command syntax is

```
prtacct file [heading]
```

The `runacct` procedure, described later in this chapter, creates several intermediary reports before compiling them into the daily report using `prdaily`. Many of these reports are accessible with the `prtacct` procedure.

Files you can use `prtacct` on in the `/usr/adm/acct/sum` directory are:

|                                |                                                         |
|--------------------------------|---------------------------------------------------------|
| <code>tacct</code>             | total <code>tacct</code> file for current fiscal period |
| <code>tacct</code> <i>mmdd</i> | <code>tacct</code> file for day <i>mmdd</i>             |
| <code>tacctprev</code>         | total <code>tacct</code> file without latest update     |

Files you can use `prtacct` on in the `/usr/adm/acct/fiscal` directory are:

|                              |                                                           |
|------------------------------|-----------------------------------------------------------|
| <code>tacct</code> <i>mm</i> | total <code>tacct</code> file for fiscal period <i>mm</i> |
|------------------------------|-----------------------------------------------------------|

Files you can use `prtacct` on in the `/usr/adm/acct/nite` directory are:

|                                 |                                                                 |
|---------------------------------|-----------------------------------------------------------------|
| <code>daytacct</code>           | total <code>tacct</code> records for this days accounting       |
| <code>disktacct</code>          | disk <code>tacct</code> records produced by disk shell          |
| <code>ctacct</code> <i>mmdd</i> | connect <code>tacct</code> records for day <i>mmdd</i>          |
| <code>ptacct</code> <i>mmdd</i> | process <code>tacct</code> records n files for clay <i>mmdd</i> |

## The `prctmp` procedure

The script `prctmp` prints login session record files; `/usr/adm/acct/nite/ctmp` is the file usually used. The command syntax is

```
prctmp [file...]
```

## Routine accounting procedures

The system uses routine accounting procedures to generate information describing what a user is doing and how often the user invokes a specific command. It also generates information on overall system usage, frequency of usage, and system resource allocation.

The system accounting lines in the files `/usr/spool/cron/crontabs/adm` and `/etc/rc` cause the system to automatically run the `startup`, `turnacct`, `ckpacct`, `dodisk`, `monacct`, and `runacct` procedures. Each of these six procedures is described in this section.

## The startup procedure

With system accounting activated, whenever the system starts up in multi-user mode (the default), `/usr/lib/acct/startup` is executed. This program has three effects:

- The `acctwtmp` program records a boot in the `/etc/wtmp` file. It uses your system name as the login name in the file.
- The `turnacct` program begins the process accounting. The `accton` program is executed, and the collected data is stored in the `/usr/adm/pacct` file. This file is later read by the reporting function and summarized in the daily and monthly reports.
- The `remove` shell procedure is executed. This procedure cleans up the temporary `pacct` and `wtmp` files that `runacct` creates.

## The chargefee procedure

You can invoke the `chargefee` shell procedure to charge a number of units to a login name. The command syntax is

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

For example, you charge login name `john` for two units for system usage. This information is written to `/usr/adm/fee` and merged with the other accounting records during the night. This information then appears in the `FEE` column in the daily report generated by `runacct`.

## The ckpacct procedure

After process accounting has begun, `crond` executes the `ckpacct` procedure every hour. This procedure checks the size of the `pacct` file. The `ckpacct` procedure begins the process of creating multiple `pacct` files when the file grows to 500 blocks.

It executes the `turnacct` command with the `switch` option (which turns the process accounting off), moves the current `/usr/adm/pacct` data to `/usr/adm/pacct./incr` (where `incr` is a number that starts with 1 and increases by

one for each additional `pacct` file that `turnacct` creates), and then turns the process accounting back on. This limits the `pacct` files to a reasonable size. If you ever need to restart `runacct`, the smaller file size makes the job easier.

Additionally, `ckpacct` checks the amount of free space left in the `/usr` filesystem; if there is less than 500 blocks free `ckpacct` turns off system accounting, and sends advisory mail to the users `root` and `adm`.

## The `dodisk` procedure

The `cron` program invokes the `dodisk` procedure to perform the disk accounting functions. The command is structured as follows:

```
/usr/lib/acct/dodisk [-o] [file...]
```

If you specify no options (the default), the procedure performs disk accounting on the files in `/etc/fstab` and `/etc/inittab`, which is a list of all file systems in the disk partition.

If you use the `-o` flag, a slower version of disk accounting by login directory is done.

The *file* specifies the name or names of the file system or file systems in which the disk accounting is done. If you use the *file* argument, disk accounting is performed on these file systems only. If you use the `-o` flag, *file* should be the names of the directories on which the file systems are mounted. If you omit the `-o` flag, *file* should be the special filenames of mountable file systems.

## The `monacct` procedure

The monthly accounting summary is another automated procedure in the accounting package. You should invoke `monacct` once each month or once each accounting period. The line in the `cron` file

```
15 5 1 * * /usr/lib/acct/monacct
```

causes `monacct` to be invoked once per month (see “Starting System Accounting” earlier in this chapter).

When run from the command line, the form of the `monacct` command is

```
/usr/lib/acct/monacct [number]
```

where *number* indicates a month or period. You can specify the week (01–52), the month (01–12), or the fiscal period, such as quarters (01–04). If you don't specify an argument, `monacct` uses the current month as the default. The `monacct` procedure creates summary files in `/usr/adm/acct/fiscal` and restarts summary files in `/usr/adm/acct/sum`.

If you want to charge users with the accounting function, you must add your own charging routines to this script.

## The `runacct` procedure

The main daily accounting procedure is `runacct`. This section covers the `runacct` command itself, the error messages it generates, and the way to recover if the procedure fails.

The `runacct` command has the following form:

```
/usr/lib/acct/runacct [mmd] [state]
```

You can use the options to restart `runacct` after a failure. They are explained later in this section.

An entry in the `/usr/spool/cron/crontabs/adm` file initiates the `runacct` procedure during nonprime hours. The `runacct` procedure automatically processes the connect, fee, disk, and process accounting files and prepares daily and cumulative summary files, which are then read by `prdaily` or used for billing purposes. When you run `monacct`, these daily reports are summarized into a monthly report then removed.

### **Restarting** `runacct`

The `runacct` procedure is designed to recognize possible errors and give warnings before terminating the process. During processing, messages are written in the `/usr/adm/acct/nite/active` file to inform the operator of successful completion of the various phases of the procedure.

Diagnostics are written into the `fd2log` (all `runacct` files are located in the `/usr/adm/acct/nite` directory unless otherwise specified). The `runacct` procedure informs you if `lock` or `lock1` exists. To prevent generation of more than one report per day, the `lastday` file keeps a record of the month and day the program was last run.

The `runacct` procedure does not damage active accounting or summary files. It records its progress by writing messages into `/usr/adm/acct/nite/active`. When it detects an error, it writes a message to the console, sends mail to `root` and `adm`, and then terminates.

The `runacct` procedure uses a series of lock files to prevent reinvocation of the accounting process until the errors have been corrected. It uses the files `lock` and `lock1` to prevent simultaneous invocation; the file `lastdate` prevents more than one invocation per day.

### **In case `runacct` fails**

If you must restart `runacct` after a failure, begin by following these steps:

- 1 Check for diagnostic error messages in the `active` *mmd* file located in the `/usr/adm/acct/nite` directory. If this file contains error messages and the lock files exist, check the `fd2log` for unusual messages.**
- 2 Fix any corrupted data files, such as `pacct` or `wtmp`. See the section “Fixing Corrupted Files,” later in this chapter, for detailed information on recognizing and fixing corrupted files.**
- 3 Remove the `lock`, `lock1`, and `lastdate` files if they are present.**

If `runacct` cannot complete the procedure for any reason (lock file encountered, error encountered, or the like), a message is written to the console (with copies sent via mail to `root` and `adm`), locks are removed, diagnostic files are saved, and the process is terminated. If you review the messages in the active file and at the console or in mail, you can determine at which point the process was stopped and why. You can then restart the process at the appropriate location and let it finish.

To make it easier to recover from errors, `runacct` is broken down into separate, restartable states. Under ordinary circumstances, the name of the state is written into `statefile` as each state is completed. The `runacct` procedure then checks `statefile` to determine what has been done and what state to process next.

States are executed in the following order:

|            |                                                                                                                                                                 |
|------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------|
| SETUP      | Moves active accounting files into working files.                                                                                                               |
| WTMPFIX    | Verifies the integrity of the <code>wtmp</code> file and corrects date changes if necessary.                                                                    |
| CONNECT1   | Produces connect session records in <code>ctmp.h</code> format.                                                                                                 |
| CONNECT2   | Converts <code>ctmp.h</code> format files into <code>tacct.h</code> format.                                                                                     |
| PROCESS    | Converts process accounting records into <code>tacct.h</code> format.                                                                                           |
| MERGE      | Merges the connect and process accounting records.                                                                                                              |
| FEES       | Converts the output of <code>chargefee</code> into <code>tacct.h</code> format and merges it with the connect and process accounting records.                   |
| DISK       | Merges the disk accounting records with connect, process, and fee accounting records.                                                                           |
| MERGETACCT | Merges the daily total accounting records in <code>daytacct</code> with the summary total accounting records in the file <code>/usr/adm/acct/sum/tacct</code> . |
| CMS        | Produces the command summaries.                                                                                                                                 |
| USEREXIT   | You can include customized accounting procedures here.                                                                                                          |
| CLEANUP    | Cleans up the temporary files and exits. <code>COMPLETE</code> appears in this file when <code>runacct</code> is finished.                                      |

The `runacct` procedure begins processing with the next state in `statefile`. If you want to begin processing at another state, include the desired state on the command line to designate where processing should begin. You must also include the argument `mdd`, specifying the month and day for which `runacct` should rerun the accounting process.



For example,

```
nohup runacct 0601 2>>/usr/adm/acct/nite/fd2log&
```

restarts the accounting process for June 1 (reporting data for May 31) at the next state in `statefile`. If you enter

```
nohup runacct 0601 MERGE 2 >>/usr/adm/acct/nite/fd2log&
```

then `runacct` restarts on June 1 at the merge state.

Normally, it is not a good idea to restart `runacct` in the setup state. Instead, run `SETUP` manually and restart by giving the command

```
runacct mmdd WTMPFIX
```

If `runacct` failed in the process state, be sure to remove the last `ptacct` file, because it will not be complete.

### **Error messages**

The `acct.cms -a` command produces a `core` file in `/usr/adm/acct` each day that system accounting runs.

The `runacct` program produces a core dump in `/usr/adm/acct` if given filenames contain a period or an underscore.

If `runacct` is terminated, error messages are written into the active `mmdd` file in the `/usr/adm/acct/nite` directory. If this file and the lock files exist, check `fd2log` for unusual messages.

The following are some common error messages and possible solutions. The list is by no means complete.

```
ERROR: acctg already run for date:
```

```
check/usr/adm/acct/nite/lastdate
```

Today's date is the same as the last entry in `lastdate`; remove the last entry in `lastdate`.

ERROR: connect acctg failed:  
check /usr/adm/acct/nite/log  
The acctcon1 program encountered a bad wtmp file. Use fwtmp to correct it.

ERROR: Invalid state,  
check /usr/adm/acct/nite/active  
The statefile is probably corrupted. Check it and read the active file before restarting.

ERROR: locks found, run aborted  
The files lock or lock1 were found in the /usr/adm/acct directory. You must remove them to restart runacct.

ERROR: Spacct?.*mmdd* already exists  
File setups probably have already been run. Check the status of files and run setups manually.

ERROR: turnacct switch returned rc= ?  
Check the integrity of turnacct and accton. The accton program must be owned by root and have the setuid bit set.

ERROR: /usr/adm/acct/nite/wtmp.*mmdd* already exists. Run setups manually.  
File setups have probably already been run. Check the status of files and run setups manually.

ERROR: wtmpfix errors see /usr/adm/act/nite/wtmperror  
The wtmp file is corrupted. Use fwtmp to correct it.

## Fixing corrupted files

When it is necessary to restart `runacct`, you may need to recreate some of the files before proceeding. You can ignore some, and you can restore others from backups. Some files, however, *must* be fixed.

**Fixing `wtmp` errors** If the date is changed while the system is in multi-user mode, a set of date change records is written into `/etc/wtmp`. The `wtmpfix` program is designed to modify the time stamps in the `wtmp` files when this happens. If there has been a combination of date changes and reboots, the `wtmpfix` program might not work, causing `acctcon1` to fail.

If this happens, you should make the following adjustment:

**1 Issue the following commands to convert the `wtmp` file to an editable file:**

```
cd /usr/adm/acct/nite
fwtmp < wtmpmmdd > xwtmp
```

**2 Edit the temporary file to delete the corrupted records, or delete all records from beginning up to the date of change.**

```
TextEditor xwtmp &
```

**3 Convert the editable file back to `wtmp` format.**

```
fwtmp -ic < xwtmp > wtmpmmdd
```

If you can't fix the `wtmp` file, create a null `wtmp` file, which will prevent connect time from being charged incorrectly. The `acctprcl` procedure is not able to determine which login used a specific process; it will charge the process to the first login in that user's password file.

**Fixing `tacct` errors** If you are using the accounting system to charge users for system usage, you must maintain the integrity of the `tacct` file in the `/usr/adm/acct/sum` directory.

If `tacct` records have negative numbers, duplicate user IDs, or a user ID of 65,535, the file may be corrupted. First, check `sum/tacctprev` with `prtacct`. If it looks all right, patch up the latest `sum/tacct` *mmdd*, then recreate `sum/tacct`.

A sample patchup follows:

**1 Issue the following commands to convert the `tacct` file to an editable file:**

```
cd /usr/adm/acct/sum
acctmerg -v < tacct mmdd > xtacct
```

**2 Edit the temporary file to delete the corrupted records.**

```
TextEditor xtacct &
```

**3 Convert the editable file back to `tacct` format.**

```
acctmerg -i < xtacct > tacct mmdd
```

**4 Remove the bad records and write duplicate UID records to another file.**

```
acctmerg tacctprev < tacct mmdd > tacct
```

◆ **Note** You can recreate the total fiscal-period accounting file `sum/tacct` by merging all the `tacct` *mmdd* files (the `monacct` procedure does this). ◆

# Stopping system accounting

There are three ways to turn off the accounting function, two temporary, one permanent.

- To turn off system accounting temporarily, use the `turnacct` command. For example, the command

```
turnacct off
```

turns the accounting function off until a `turnacct on` command is issued, or the system is restarted.

- The `shutacct` command is usually used during system shutdown. The syntax of the command is

```
shutacct [reason]
```

When invoked with the option a *reason* record is appended to `/etc/wtmp`.



- To permanently turn off the accounting function, reverse the actions taken in steps 2 and 3 of the section “Starting System Accounting,” earlier in this chapter. That is, comment out the accounting invocation lines in the `/etc/rc` and `/usr/spool/cron/crontabs/adm` files.



# Appendix E: Using the $d_p$ Utility

Manipulating slice numbers / E-3

Assigning permanent slice numbers / E-6



Occasionally you may want to have more partitions than Hard Disk SC Setup can provide (six for a disk with a root file system, four otherwise) or you may want to create partitions on your disk of type Misc UNIX. In these cases you must use the disk partitioning utility `dp`.

This appendix gives you procedures to use in these instances.

◆ **Note** On the boot disk, the types Root, Root&Usr, Usr, Swap, and Autorecovery are reserved. The Swap type should be reserved for swapping on all disks. (Table E-1 shows how A/UX maps the various partition types to unique slice numbers.) In the context of the `dp` utility, *UFS* refers to the `/usr` file system tree, not the Berkeley File System. ◆

**Table E-1** A/UX partition types available

| <b>A/UX partition type P</b> | <b>Reassigned A/UX slice number</b> |
|------------------------------|-------------------------------------|
| A/UX Autorecovery            | N/A                                 |
| A/UX Root&Usr                | 0                                   |
| A/UX Root                    | 0                                   |
| A/UX Swap                    | 1                                   |
| A/UX Usr                     | 2                                   |
| Free A/UX                    | 3                                   |
| Free A/UX                    | 4                                   |
| Free A/UX                    | 5                                   |
| Free A/UX                    | 6                                   |
| Misc A/UX                    | N/A                                 |

## Manipulating slice numbers

The following procedure leads you through the necessary steps to locate partitions with slice numbers that are not unique, or with missing slice numbers, and to make any necessary changes. Before following these procedures, use Hard Disk SC Setup to create and mount file systems with automatically mapped slice numbers. You can have up to six local partitions in the startup disk and up to four on other disks without using `dp` (four on other disks because types Root (slice 0) and Usr (slice 2) are reserved for the boot disk only).



- ▲ **Warning** The partition type A/UX Autorecovery is reserved and must never be used for your own use. ▲

## 1 **Start up A/UX, if you have not done so already.**

Choose Restart from the Special menu. Let the system startup from your installed version of A/UX.

## 2 **Backup all files on your disk.**

Before making any change to disk partitions, you should have a set of reliable backups.

## 3 **Obtain the index numbers for any partitions with identical names, and if present, change the duplicates to unique names.**

Enter the following command:

```
echo p | dp -q /dev/rdisk/c x d0s31 | egrep "Index|Name"
```

where  $x$  is the SCSI address of the disk under investigation. The `dp` utility responds with a report showing the index number and name of each partition previously allocated on that disk using the Apple Hard Disk SC Setup program. Make note of these values because you will use them in a subsequent step.

The names shown by `dp` are those shown in the Details window of Hard Disk SC Setup.

If duplicate names exist, you must assign unique names by using the editing capabilities of `dp`. If no duplicates exist, proceed to step 4.

Replacing the drive number  $n$  with the SCSI ID number of the new drive, enter the following command:

```
dp /dev/rdisk/c n d0s31
```

The `dp` utility then prompts you for a command. For example, if you are partitioning a disk with SCSI ID number 6, the following message appears:

```
"/dev/dsk/c6d0s31" n partitions, m allocated [unknown sizes]
Command?
```

◆ **Note** If you get a different response, press the lowercase `q` to quit `dp`. Reenter the `dp` command just given, making sure you specify the correct SCSI ID and type the command correctly. ◆

Repeat the following series of substeps as many times as necessary to eliminate duplicate partition names.

- a. Replacing the value *x* with the index number of one of the identically named partitions, enter `cx`.  
You are prompted to identify the attribute field that you wish to change:  
DPME Field?
- b. Enter `n` to begin changing the partition's *name* field. Now you are prompted for a name for the partition:  
Name [*Old-name*]:
- c. Give the partition a name that is unique and that does not contain any embedded spaces. If `A/UX_Partition` has not already been used, then you can enter `A/UX_Partition`.  
Again you are prompted for the attribute field that you wish to change  
DPME Field?
- d. Enter `q` to return to the `dp` utility's first menu level. You will see the command prompt `Command?`. If you have no more partition names that are not unique, you should save all the cumulative changes and quit `dp` by entering `w`.  
The root command prompt should reappear on the screen. Otherwise, if you still have any remaining partitions without unique names, return to substep a.

#### 4 **If you have partitions named other than Misc A/UX, use `dp` to associate slice numbers with these unmapped partitions.**

You can assign slice numbers permanently using `dp`; see "Assigning Permanent Slice Numbers" in the following section.

#### 5 **If you want to run file system consistency checks against the new partitions each time you restart, then the startup files need to be altered.**

See "Multiple File Systems and `fsck`" in Chapter 8.

# Assigning permanent slice numbers

To assign a permanent slice number to a partition, work either from A/UX Startup or at the A/UX command line. When you work within A/UX, the disk that you want to assign a permanent slice number should *not* be in use. For additional information on `dp`, see `dp(1M)` in *A/UX System Administrator's Reference*.

Follow these steps to assign a permanent slice number to a disk.

## 1 Determine which slice numbers have already been assigned.

Enter the command:

```
dp /dev/rdisk/c1d0s31
```

where `s31` stands for the entire disk. The system responds with the total number of partitions, the number of partitions allocated, and the total number of blocks on the disk; for example:

```
"/dev/dsk/c1d0s31" 9 partitions, 9 allocated 156369 blocks
followed by the Command? prompt.
```

## 2 Request the current information about a specific partition.

For example, if you want information about partition 7, enter the command:

```
p 7
```

where `p` prints the following general information about the partition:

```
DPM Index: 7
Name: "Unreserved 1", Type: "Apple_UNIX_SVR2"
Physical: 2 @ 156366, Logical: 2 @ 0
Status: valid alloc in_use not boot
 read write
Slice 3
Regular UNIX File System (1)
Cluster: 0 Type: FS Inode: 1
Made: [0] Wed Dec 31 16:00:00 1969
Mount: [0] Wed Dec 31 16:00:00 1969
Umount: [0] Wed Dec 31 16:00:00 1969
```

Some of the components of this information are:

No AltBlk map

Tells whether or not an alternate block map exists for the partition.

DPM Index: 7

The Disk Partition Map index number for partition 7 is the seventh record.

Name: "Unreserved 1", Type: "Apple\_UNIX\_SVR2"

Gives the name and type of the partition.

Physical: 2 @ 156366, Logical: 2 @ 0

Shows the physical and logical locations of the partition. In this example, the partition is two physical blocks long and starts at block 156366. For the user, the partition is two blocks long and starts at block 0.

```
Status: valid alloc in_use not boot
 read write
```

Gives general information about the partition. Of importance to you is whether the disk can be read from or written to.

The following information that is printed is specific to A/UX file systems.

Slice 3

If you try to open Slice 3, you will be connected to the part of the disk that is described by the partition entry; in this case, partition 7.

No AltBlk map

Tells whether or not an alternate block map exists for the partition.

### 3 **Enter the `change`, or `c`, command at the `Command?` prompt to modify the partition map for a partition:**

For example, enter

```
c 7
```

to modify the partition map entries for partition 7. You are prompted to enter the field in the Data Partition Map Entry (DPME) that you want to modify.

**4 Enter b for block zero block (BZB).**

```
DPME Field? b
```

BZB is the name for a structure containing A/UX-specific information about the partition. You are then prompted to enter the field in the BZB that you want to modify.

**5 Enter s for slice number.**

```
BZB Field? s
```

**6 Enter the new slice number plus one in response to the Slice number prompt (the system subtracts one from the number you enter).**

```
Slice number + 1 [8]: 11
```

The current slice number is shown in brackets. In this example, the number 11 is entered, the new slice number is 10. You are then prompted to enter the next slice number to be modified.

**7 Enter p to display the new information about the BZB field.**

```
BZB Field? p
```

The following information is displayed:

```
Slice 10
Regular UNIX File System (1)
Cluster: 0 Type: FS Inode: 1
Made: [0] Wed Dec 31 16:00:00 1969
Mount: [0] Wed Dec 31 16:00:00 1969
Umount: [0] Wed Dec 31 16:00:00 1969
No AltBlk map
```

**8 Enter q at the BZB and DPME field prompts to stop modifying the partition map entries:**

```
BZB Field? q
```

```
DPME Field? q
```

You then return to the `Command?` prompt.

9 **Enter `w` at the `Command?` prompt to write the permanent slice number to disk.**

10 **Enter `Q` at the `Command?` prompt to exit the program.**

The slice number that you assigned to the disk remains until you change it again using the `dp` program.