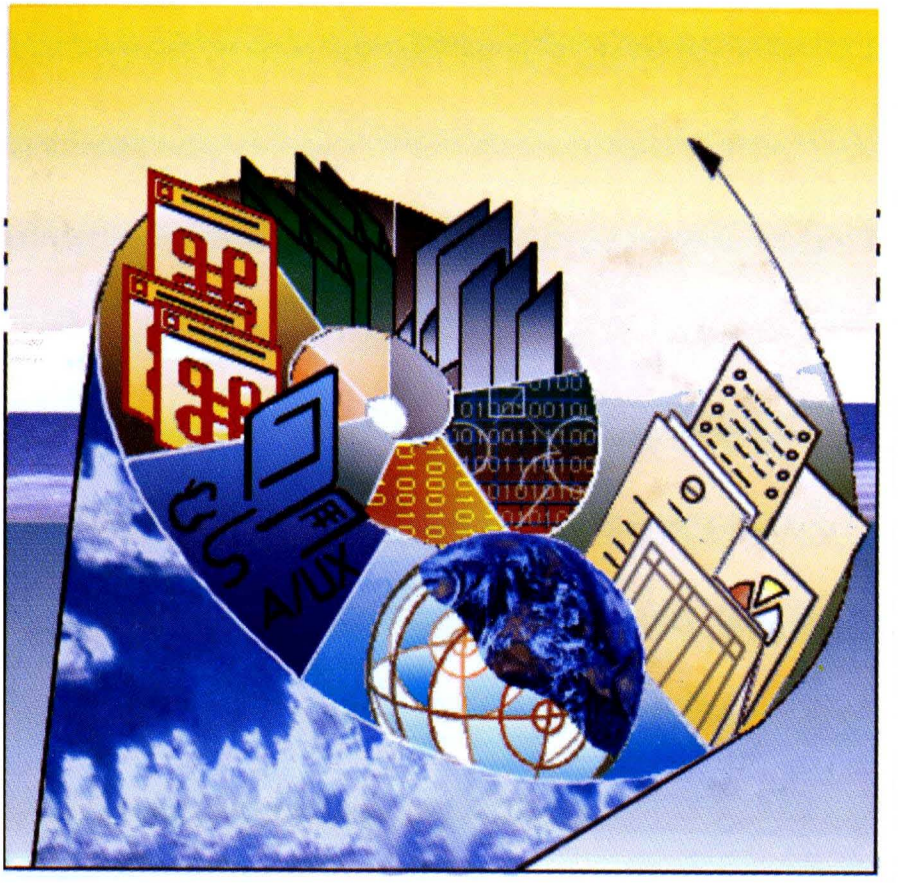
A/UX® Local System Administration



A/UX® Local System Administration

030-0762-A

APPLE COMPUTER, INC.

Apple Computer, Inc.
20525 Mariani Ave.
Cupertino, California 95014
(408) 996-1010

Simultaneously published in the United States and Canada.

030-0762-A

LIMITED WARRANTY ON MEDIA AND REPLACEMENT

If you discover physical defects in the manual or in the media on which a software product is distributed, Apple will replace the media or manual 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 THIS MANUAL, 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 reviewed this manual, **APPLE MAKES NO WARRANTY OR REPRESENTATION, EITHER EXPRESS OR IMPLIED, WITH RESPECT TO THIS MANUAL, ITS QUALITY, ACCURACY, MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE. AS A RESULT, THIS MANUAL IS SOLD "AS IS," AND YOU, THE PURCHASER, ARE ASSUMING THE ENTIRE RISK AS TO ITS QUALITY AND ACCURACY.**

IN NO EVENT WILL APPLE BE LIABLE FOR DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES RESULTING FROM ANY DEFECT OR INACCURACY IN THIS MANUAL, even if advised of the possibility of such damages.

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.

Contents

Figures and tables

Preface

As an A/UX® system administrator, you will perform these tasks to ensure that the system runs properly:

- Starting up and shutting down the system
- Adding, modifying, and removing user accounts
- Backing up the system
- Managing peripheral devices
- Locating and resolving inconsistencies in the file system
- Monitoring user and system activity and disk space
- Setting up and maintaining communication channels with other computer systems
- Troubleshooting and solving system problems

This guide consists of the following chapters:

- Managing the A/UX System: An Introduction
- System Startup and Shutdown
- User and Group Administration
- Backing Up Your System
- Preparing an Apple® Hard Disk SC for A/UX
- Managing Disks
- Managing Other Peripheral Devices
- Checking the A/UX File System: `fsck`
- System Accounting Package
- System Activity Package
- Troubleshooting

Who should read this guide

This guide is for both new and experienced system administrators. Except for communicating with other systems, all of the procedures discussed in this guide affect a single computer and are considered **local system administration.** For information on setting up and administering networks, see *A/UX Network System Administration.*

How to use this guide

After reading the introductory chapter, you should become familiar with starting up and shutting down the system. To begin your work as system administrator, learn how to set up user accounts and to back up the system. New system administrators should at least skim the rest of the manual sequentially to gain an overall picture of what duties they are to perform. You can read about how to perform a specific task as the need arises. Experienced administrators can use the manual as a reference.

What you should already know

As system administrator, you need to have a basic knowledge of how to use A/UX, whether or not you have had previous system administrative experience.

Conventions used in this guide

A/UX guides follow specific conventions. 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	ESCAPE	SHIFT
COMMAND	LEFT ARROW	TAB
CONTROL	RETURN	UP ARROW
DOWN ARROW	RIGHT ARROW	

For example, suppose you enter

```
Applee
```

instead of

```
Apple
```

To erase the additional *e*, you would position the cursor (or insertion point) to the right of the word and press the DELETE key once.

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 an entry 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.

Term	Action
Choose	Activate a command in a menu. To choose a command from a pull-down menu, click once on the menu title while holding down the mouse button, and drag down until the command is highlighted. Then release the mouse button.
Click	Press and then immediately release the mouse button.
Drag	Position the pointer on an object, then press and hold down the mouse button while moving the mouse. Release the mouse button when the object reaches the desired position on the screen.
Enter	Type the letter or letters and press the RETURN key.
Press	Type a *single* key *without* pressing the RETURN key. Or position the pointer on an object and hold down the mouse button.
Select	Position the pointer on a selectable object and click the mouse button.
Type	Type an entry *without* pressing 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 the 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 type `date` and press RETURN, you'll see something like this:

```
Tues Oct 17 17:04:00 PDT 1989
```

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

All A/UX 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.

Font styles

Words that you must replace with a value appropriate to a particular set of circumstances appear in *italics*. For example, if you see

`cat` *filename*

replace the italicized word with the name of the 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.

A/UX command syntax

A/UX commands follow a specific command syntax. A typical A/UX command has this form:

`command` [*flag-option*] [*argument*]...

The following table outlines the elements of an A/UX command.

Element	Description
`command`	The command name.
flag-option	One or more optional arguments that modify the command. Most flag options have the form [-*opt*...], where *opt* is a letter representing an option. Most commands have one or more flag options.
argument	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 an argument that can be repeated any number of times.

For example, the `wc` command is used to count lines, words, and characters in a file. Here is the full syntax for that command, including all possible flag options and the optional argument *name.*

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

Thus, you can enter

```
wc -w /Priscilla
```

to count all of the words in the file `/Priscilla`, where `wc` is the name of the command, `-w` is the flag option that instructs the command to count all of the words in the file, and the optional argument `/Priscilla` is the file to be searched.

Command reference notation

A/UX Command Reference, A/UX Programmer's Reference, and *A/UX System Administrator's Reference* contain references for commands, programs, and other related information. Material is organized within these references by section numbers. The standard A/UX cross-reference notation is

cmd (sect)

where *cmd* is the name of the command, file, or other facility; *sect* is the section number where the entry resides.

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

For example,

`cat(1)`

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

References can be also called up on the screen. Use the `man` command to display pages from reference manuals, known as manual pages, directly on the screen. 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 shell prompt, or type `q` at any time to return immediately to your shell prompt.

Cross-referencing

An A/UX guide often refers to information discussed in another guide in the suite. The format for this type of cross-reference is "Chapter Title," *Name of Guide.*

For a complete description of A/UX guides, see *Road Map to A/UX.* This guide contains descriptions of each A/UX guide, part numbers, and ordering information for all the guides in the A/UX documentation suite.

Additional conventions

Three shortcut programs—`adduser`, `fsentry`, and `setport`—provide quick and easy ways to set up user accounts and peripheral equipment. If and when you want to learn to perform these actions using standard A/UX, or need an in-depth discussion of administration issues, refer to the appropriate section of this manual. When Commando dialog boxes are also available for a task described in this manual, the **racer icon** in the left margin alerts you to this fact. For complete instructions on using the adduser, fsentry, and setport Commando dialogs, refer to *Setting Up Accounts and Peripherals for A/UX.*

Using Commando

Chapter 4, "Using Commando," of *A/UX Essentials* introduced Commando dialogs. You should experiment with the Commando interfaces to common system administration commands. For example, if you have previously worked with SVFS file systems, the `newfs` command is probably new to you. Enter `newfs` followed by COMMAND-K at the A/UX command line to display the Commando dialog.

For a description of how to use the newfs Commando dialog, see "Using `newfs`" in Chapter 5.

Chapter 1 Managing the A/UX System: An Introduction

This book combines theory and practice to help you understand what to do as well as why. If you are not an experienced system administrator, here are some suggestions before you begin:

- Log in as the superuser (root) only when absolutely necessary. The superuser has special privileges, and you can perform functions that you can't perform if you are logged in as a normal user. Thus as a normal user you have some protection against making mistakes that might affect the operation of your system. If you make mistakes as the superuser, you can damage your system.
- While you're logged in as the superuser, record everything you do in a system administrator's log. This log should contain an exact description of what you do, why you do it, when you do it, and what effect it has.
- Make backups of all important system files before you change them. For example, if you are about to modify the `/etc/inittab` file, first enter the command

  ```
  cp /etc/inittab /etc/inittab.old
  ```

  Then, if your modifications prove unworkable, it is an easy matter to return the system to its prior state with the command

  ```
  mv /etc/inittab.old /etc/inittab
  ```
- Use `cron` to automate system administration routines or to remind yourself to do these duties. See Chapter 6, "Managing Disks," Chapter 9, "System Accounting Package," and Chapter 10, "System Activity Package." Also see `cron`(1M) in *A/UX System Administrator's Reference* and `crontab`(1) in *A/UX Command Reference.*

- Use the `find` command to locate large dormant files and directories. Large unused files are often not needed and consume valuable storage space. When you find such files, you can either remove them (for example, if they are core files) or truncate them to zero length if they are log files. (**Core files** are system memory images of damaged programs used by system programmers for troubleshooting.) Transfer files to tape or floppy disk archive if you are even remotely likely to want to use them again in the future, or compress them to a smaller size. See Chapter 4, "Backing Up Your System," or refer to `find`(1) in *A/UX Command Reference.*
- Watch for files that grow (in general, any file containing the letters `log` or `LOG` as part of its name). The system appends information to these files, and they can grow to excessive size. You should regularly delete or truncate these files after backing them up. See Chapter 6, "Managing Disks."
- Make frequent backups of your files. 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 4, "Backing Up Your System."
- If you are administering a system with multiple users, choose low-activity times to perform potentially disruptive tasks. Early morning and late evening are good choices. Check to see who is logged in before you begin; if anyone is actively running processes, notify them with the `wall` command. See `wall`(1M) in *A/UX System Administrator's Reference.*
- If a catastrophe occurs:
  1. Think carefully about what happened.
  2. Plan your next action before actually doing it.
  3. Anticipate the consequences of implementing your plan.
  4. Act.
  5. Write down in your system log what happened, what you did, and how the system responded.

One of the worst things you can do in the event of a system catastrophe, such as a loss of important data, is to act without first considering the consequences of your actions.

Administrative logins on the A/UX system

The standard distribution of the A/UX® system contains one normal user login account—`start`—whose home directory is `/users/start`. This login account is provided to give new users a place to store files used in the introductory tutorials; see *A/UX Essentials.*

There is also the Guest account, which can be used when the system is first installed, and to enable a guest to have quick access to run Macintosh® programs. The system administrator should create an account for each user who will normally need access to the system, and may remove the Guest account for added security.

Administrative logins are used by system administrators or by programs to perform specialized system tasks. The administrative logins, which are found in the `/etc/passwd` file, are as follows. (Note that `root` is the only account you will use; the other administrative logins are used by programs.)

`root` — Login for the superuser. As shipped, the home directory is `/`, and the shell is `/bin/sh`. The home directory may be changed to `/root` in the `/etc/passwd` file. The shell may also be changed to `/bin/csh` or `/bin/ksh` in the `/etc/passwd` file, if desired. Avoid remotely mounted home directories or unusual shells.

`daemon` — The owner of certain noninteractive background processes that handle persistent system services, such as network communication and the `lpr` print spooler.

`bin` — The owner of most normal commands and the system directories in which those commands are stored.

`sys` — The owner of certain system files, such as `/etc/zoneinfo` and the files used by `autorecovery`.

`adm` — The owner of most system accounting programs and directories, in particular `/usr/adm`. See Chapter 9, "System Accounting Package."

`nuucp` — The login assigned to incoming `uucp` requests. See *A/UX Network System Administration* for more information.

`uucp`	The owner of programs and directories associated with the UUCP communications package.
`lp`	The owner of commands and processes associated with the `lp` line printer spooling and printing package. See Chapter 7, "Managing Other Peripheral Devices," for more information on the `lp` system.
`ftp`	The owner of files and directories associated with `ftp`, a file transfer program. See *A/UX Communications User's Guide* for a description of the `ftp` command.
`nobody`	Assigned as the default login for remote root access under NFS. See *A/UX Network System Administration* for further information.
`SVFS`	A login that allows users to see who is currently logged in to the system without first logging in themselves. There is no password for this login. The startup program is simply `/bin/who`.

Administrative groups

Similar to administrative logins, **administrative groups** help you perform specialized system tasks and are provided in the system as shipped. An administrative group lets you avoid running a program as the root user by instead running it with the group ID of the group provided for this special purpose. This reduces the security risk inherent in running programs as the root user.

The administrative groups provided in A/UX are `sys`, `bin`, `root`, `daemon`, `adm`, `uucp`, `lp`, and `mail`.

The complete contents of A/UX

A/UX consists of an enormous number of files. If you wonder exactly what those files are, you can find the answer in the complete list of distributed A/UX files. This list is provided in a file called `/FILES`, which gives the full pathname of every A/UX system file along with a short description of its use. The list is a valuable resource when you want to look up the purpose of a particular system file quickly.

File systems: UFS versus SVFS

The root A/UX 2.0 file system is set up as a Berkeley File System, usually referred to in this guide as **UFS,** but in some cases as the **BSD** or **4.2** file system. For backward compatibility with A/UX 1.0, the System V file system, usually referred to here as **SVFS,** but in some cases as the **5.2** file system, continues to be supported. UFS has the advantages of faster performance and longer file and directory names, as well as enabling you to make file systems more easily.

Both file systems employ the following utilities: `fsck`, `fsdb`, `mount`, and `umount`. To indicate to the command which file system is intended for commands that are used with both, a type parameter , `-T`, has been added to the utilities. For example, the syntax

```
fsck -T5.2 /dev/dsk/c0d0s0
```

requests that `fsck` check an SVFS file system. The syntax

```
fsck -T4.2 /dev/dsk/c0d0s0
```

refers to a UFS file system. The `fsck` utility, however, can determine which file system it is commanded to repair, so the `-T` option can be omitted.

For a partial list of the commands and utilities that access the two file systems, enter

```
ls -l /etc/fs/ufs
```

or

```
ls -l /etc/fs/svfs
```

Now that you have been introduced to system administration, let's see how the A/UX system starts up and shuts down.

Chapter 2 **System Startup and Shutdown**

This chapter explains how to start up and shut down the A/UX operating system that runs on the Macintosh-II family and Macintosh SE/30 computers. Although these procedures are largely automatic, they may require your supervision or intervention. Procedures that can be changed to suit local needs, such as setting the time and message of the day, run levels, and kernel parameters, are discussed in "Customizing Your System," later in this chapter.

This chapter also describes A/UX Startup, a Macintosh program that starts up A/UX. A/UX Startup is a command interpreter with a command language and environment similar to, but simpler than, that of the Bourne shell. You will use A/UX Startup to run a set of A/UX troubleshooting utilities when you need to work outside of the A/UX Operating System. Other topics covered include:

- What to do when your system freezes, or if you have a power failure or an emergency.
- How to change startup devices.
- How to set startup applications.
- How to set initial processes.

Overview of system startup and shutdown

When you first turn your system's power on after installation, it starts up in the Macintosh environment and stays in this environment until you select the A/UX Startup icon, which resides on the MacPartition disk. A/UX Startup is the Macintosh program that starts, or boots, A/UX. The A/UX boot sequence then launches the A/UX kernel. When the kernel is launched, the system enters the A/UX environment and stays there until you shut down the system.

Figure 2-1, "Overview of System Startup and Shutdown," shows the division of the startup procedure between the two operating systems. Refer to this figure during the following discussion of A/UX startup and shutdown as a quick visual guide to finding where you are in the procedure.

Starting up the system

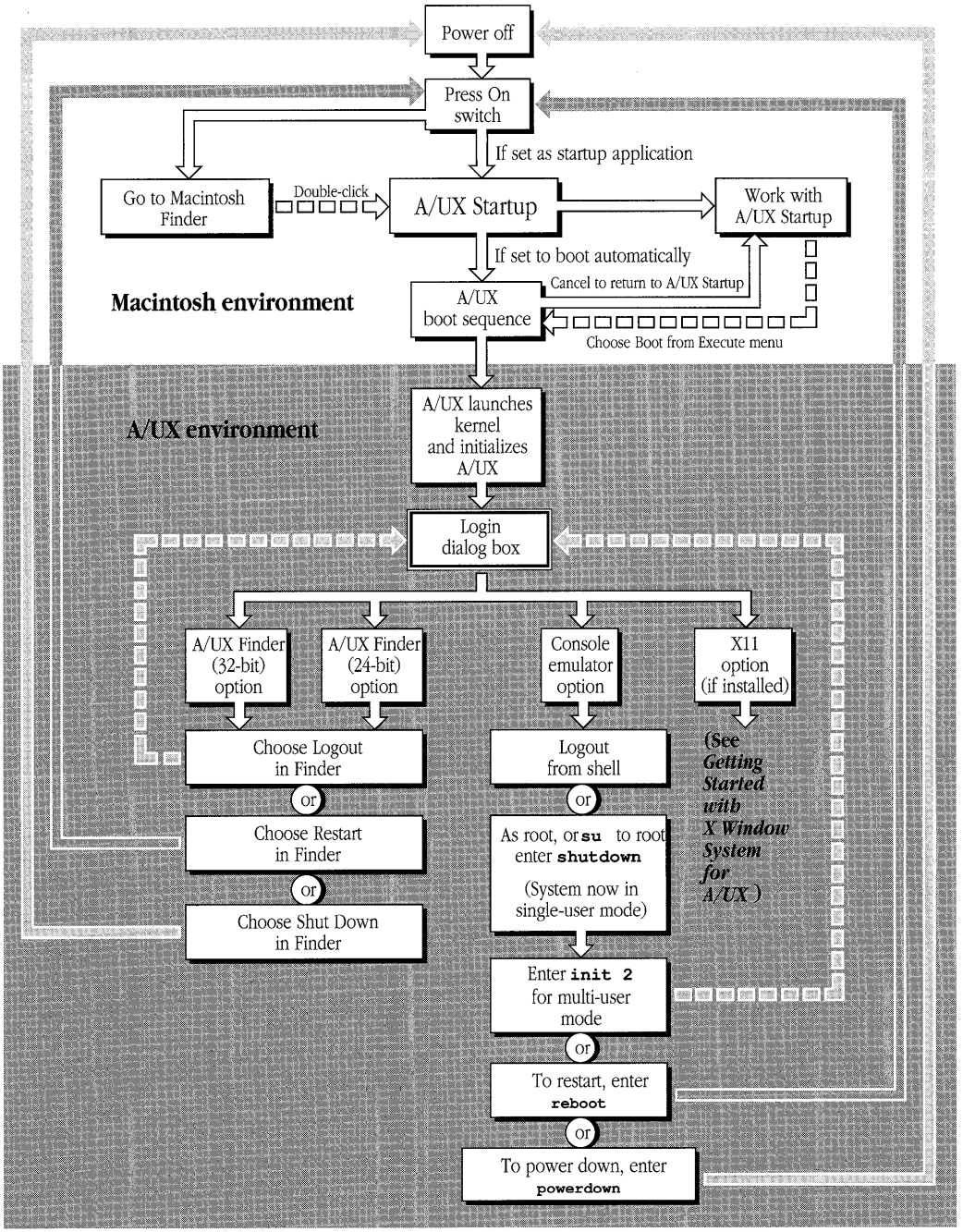
This section assumes that you have already followed the instructions for setting up your Macintosh computer given in the *A/UX Installation Guide* and have become familiar with the A/UX basics presented in *A/UX Essentials*.

Once the computer is set up, you are ready to boot the operating system. Note that an A/UX system starts up from the hard disk that contains A/UX Startup, unless a floppy disk containing a Macintosh System Folder is inserted in the internal drive.

First, turn on the power. A few seconds later, the screen lights up and the Welcome to Macintosh message appears briefly before you enter the Macintosh Finder™. On the desktop the MacPartition disk icon appears. As shown in Figure 2-1, you can either work in the Macintosh Operating System or double-click on the A/UX Startup icon in MacPartition to work with A/UX.

- *Note:* If you *always* want to work with A/UX, you can bypass the Macintosh OS by setting A/UX Startup options in the Preferences menu to automatically launch A/UX whenever you restart the system. See "A/UX Startup Menus," later in this chapter, for details on setting these options.

■ **Figure 2-1** Overview of system startup and shutdown



After A/UX Startup is selected, either by your double-clicking on the A/UX Startup icon from the Macintosh Finder, or by running automatically as the startup application, you have two choices (see Figure 2-1). You can work with A/UX Startup itself, or boot the A/UX Operating System from A/UX Startup.

The booting procedure can be done in one of two ways:

- By entering `boot` at the A/UX Startup prompt, or choosing Boot from the Execute menu in the A/UX Startup window
- By setting A/UX Startup to automatically boot A/UX after you turn on the system

Procedures for the first way are given in "Booting from A/UX Startup," in the following section. If you decide to boot A/UX Startup automatically, see "A/UX Startup Window" and "A/UX Startup Menus," later in this chapter.

Booting from A/UX Startup

Once you are in A/UX Startup, you can use its A/UX system-like utilities to work with A/UX from the Macintosh OS (Operating System). For example, if you are unable to launch the A/UX kernel, you can use A/UX Startup's subset of A/UX commands for troubleshooting. For the purpose of describing how to start up A/UX, however, we'll ignore this feature for now.

Other than setting A/UX Startup as the startup application that automatically boots A/UX, the fastest method of booting is to choose Boot from the A/UX Startup Execute menu, or to press COMMAND-B. Another way is to enter

```
boot
```

at the prompt.

If the file system was not cleanly unmounted when the system was last shut down, the `boot` command runs `fsck` on the root file system as the first phase of the boot sequence.

The boot sequence

Figure 2-2 shows the sequence of the six startup screens, each of which represents a step of the initiation procedure. A progress bar indicating how much of the six-step initiation procedure has been completed displays in each screen.

As shown in Figure 2-1, the boot sequence begins in the Macintosh OS and moves into the A/UX environment *after* the A/UX kernel is launched. This distinction is important because as long as the system is in the Macintosh OS, you can cancel the boot sequence.

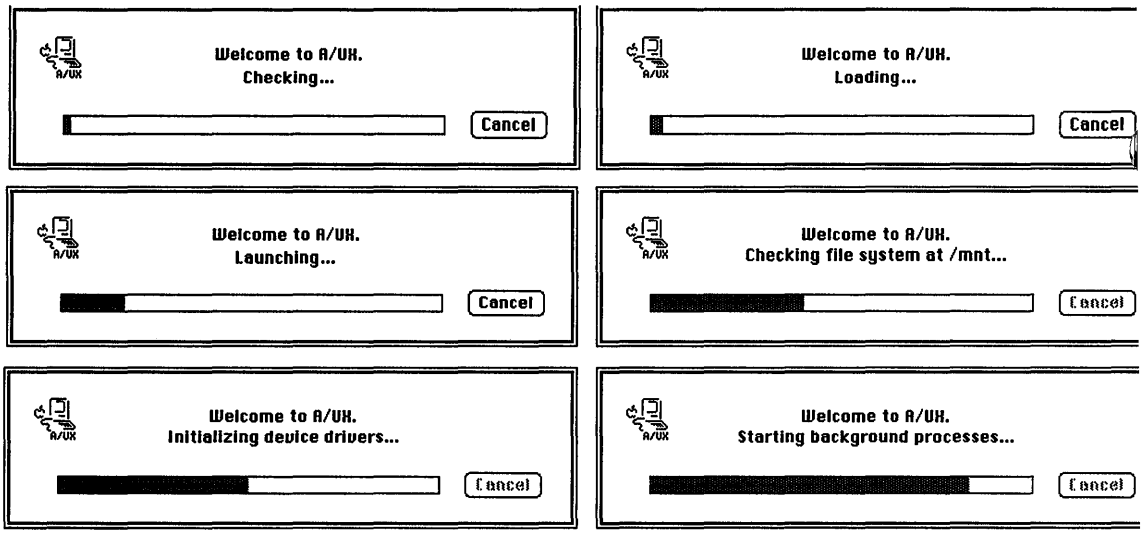
Canceling the boot sequence enables you to work exclusively within the Macintosh OS. Follow these steps:

1. **Click Cancel on the startup screen during the first several seconds of the process (while the kernel is loading and before launching starts).**
2. **Choose Quit from the File menu to return to the Finder.**
3. **Double-click on your Macintosh hard disk icon to open its files and folders.**

    - *Note:* Another way to cancel the boot sequence and enter A/UX Startup is by pressing COMMAND-period (COMMAND-.) from the Copyright dialog box.

Canceling startup always returns you to A/UX Startup.

■ **Figure 2-2** The six phases of startup



Phase 1: Checking

A/UX Startup executes a thorough check of the root file system if the file systems have been damaged.

By default, AUX Startup's `AutoRecovery` command runs `fsck` for the root file system if the root file system mount flag is on, indicating possible file system damage. See `fsck`(1M) and Chapter 8, "Checking the A/UX File System: `fsck`."

Phase 2: Loading

During this phase, A/UX Startup loads the kernel and moves the progress bar.

The boot process continues unless you click Cancel or press COMMAND-. (COMMAND-period). This action cancels the loading and places you in the A/UX Startup window with the A/UX Startup shell prompt.

◆ *Note:* The Cancel button operates during the entire loading phase.

Phase 3: Launching

When launching starts, the A/UX booting procedure cannot be stopped. (The Cancel button is dimmed, indicating that it is disabled.) During launching, A/UX Startup causes A/UX to take control of the computer. The kernel launches and is initialized. Note that the progress bar does not move during this phase. The screen, however, blinks momentarily, which is part of normal operation.

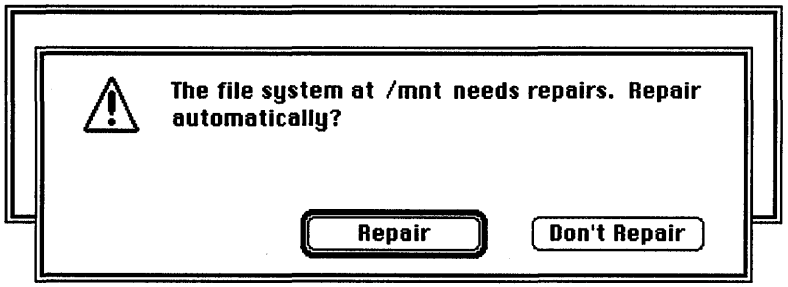
At this point, `/etc/macsysinit` launches the Macintosh environment under A/UX for the duration of the boot process. The console window can be brought to the front to view any system messages that display during the startup process as follows: Pull down the Apple menu and choose the CommandShell menu item. Another way to cause messages to be displayed is to enter the `launch -v` command in A/UX Startup. See Chapter 5, "Using CommandShell," in *A/UX Essentials* for a complete discussion of CommandShell. The `launch -v` command is discussed later in this chapter in "A/UX Startup Program."

Phase 4: Checking file systems at <mount point>

In addition to root, you can add other file systems that contain files you have created. Every file system, however, has to be checked by `fsck`.

The `/etc/bcheckrc` program runs `fsck`, checking all file systems other than root that appear in the `/etc/fstab` file with a pass number entry of 2. The specific mount points are shown in the dialog box, which enables you to monitor progress and see which file systems need to be repaired. If problems are detected for a file system, `fsck` displays a modal dialog that asks whether or not you want to proceed with repairs (see Figure 2-3). If you decide not to repair, the boot sequence continues, and you must run `fsck` for that file system at the command line after logging in, or else the file system cannot be mounted. Proper shutdowns as described in "Logging Out, Restarting, and Shutting Down," later in this chapter, are always recommended as insurance against file system damage.

■ **Figure 2-3** Dialog box displayed by `fsck`



Phase 5: Initializing device drivers

The `/etc/sysinitrc` program verifies that the root file system is clean, mounts it, and executes `autoconfig` and device driver startup scripts. If your hardware configuration has changed, `autoconfig` builds a new up-to-date kernel and reboots the system so that the new kernel is used.

Checking a file system may take longer than any other phase of the boot process, depending on whether the file system was cleanly unmounted before the last shutdown. If not, the file system should be repaired because the kernel does not allow file systems to be mounted until they have been checked and marked clean.

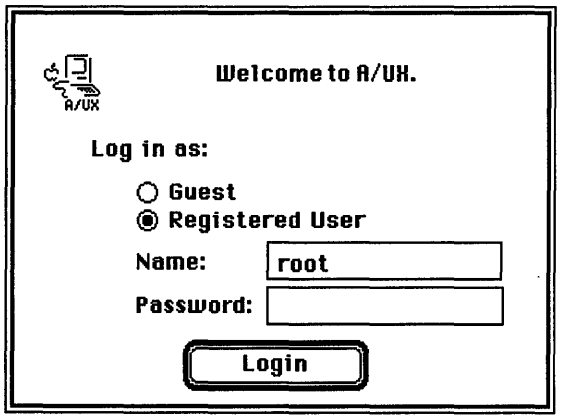
Phase 6: Starting background processes

In this last phase, `/etc/rc` mounts file systems and cleans `/tmp`, the spool directories, and log files. Then the `init` process spawns background processes, as specified by `/etc/inittab`. After this, A/UX initialization is complete.

Logging in

When the Login dialog box is displayed, enter your login name and password (see Figure 2-4).

■ **Figure 2-4** Login dialog box



As shown in Figure 2-1 (the overview diagram), there are four possible modes. A/UX Finder 32-bit mode is the default:

A/UX Finder (32-bit)
: Finder environment for 32-bit applications and CommandShell

A/UX Finder (24-bit)
: For backward compatibility with 24-bit applications

Console emulator
: For working at the A/UX command line without the Finder

X11
: If you have the X11 server installed

To select a mode, choose Change Sessions Type from the Options menu (with the Login dialog box displayed) as described in "Changing Your Session Type" in Chapter 1 of *A/UX Essentials.*

For information on the X11 option, refer to *Getting Started with X Window System for A/UX.*

Password protection

Password protection is controlled by the `/etc/passwd` file, which is discussed in Chapter 3, "User and Group Administration." If `/etc/passwd` contains an entry for a Guest user, the Guest radio button in the dialog box is enabled. If there is no entry for Guest in this file, log in for guests is prohibited. See "The `/etc/passwd` File," in Chapter 3, for suggestions on making the Guest account secure.

When anyone types a name not entered in `/etc/passwd`, the unknown user message is displayed.

If the password for that login name is incorrect, the incorrect password message is displayed.

Startup shell

If you selected A/UX Finder (32- or 24-bit mode) and entered the correct name and password, you are placed in the A/UX Finder. Select your personal folder to the right of the desktop. If you want to work in the A/UX command line (or shell), choose CommandShell from the Apple menu. The message of the day, if any, is displayed in the first CommandShell window that you open. To open a CommandShell window, choose the CommandShell menu item from the Apple menu.

In console emulator mode, the console window appears, displaying the message of the day and your shell prompt. You are placed in your home directory. (For instructions on personalizing the message of the day, see "Changing the Message of the Day," later in this chapter.)

If you already set your host name during the installation process, that host name is used by the shell prompt. (See "Renaming the System," later in this chapter, for the steps you take to change the host name.) The shell prompt also displays the name of the user, which in this example is root. Although the prompt is set by your default login script, either `.profile` for Bourne or Korn users, or `.login` for the C Shell, you can change it.

If your system's name is picasso, you will see this prompt:

```
picasso.root#
```

You are now up and running in multi-user mode. See "Changing the Startup Device and Application" and "Customizing Your System," later in this chapter for changes you can make to the booting sequence. If you want to bring the system into single-user mode automatically, see "Single- and Multi-user Modes," later in this chapter.

Logging out, restarting, and shutting down

Refer to Figure 2-1 for a quick visual guide to restarting and logging out of A/UX. From the A/UX Finder options, you will choose Logout, Restart, or Shut Down from the Special menu. Choosing Logout returns you to the Login dialog box from which you or another user can log in. This action causes the Finder to close all open Macintosh applications before it exits.

Choosing Restart closes all open Macintosh applications, unmounts the file systems, then restarts the Macintosh. You can then either enter the Macintosh Finder or A/UX Startup, or begin the automatic A/UX boot sequence, as determined by the preference set in the A/UX Startup Preferences menu (see "Preferences Menu" later in this chapter).

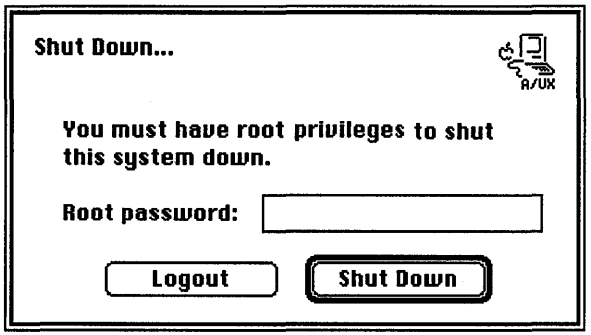
Shutting down the computer from the Finder

Follow these steps to shut down the computer:

1. **Choose Shut Down from the Special menu with either the Finder or the Login dialog box displayed.**

   You see the Shut Down dialog box, shown in Figure 2-5.

■ **Figure 2-5** The Shut Down dialog box



2. **Enter the root password in the appropriate field.**

   If you logged in as the root user, you are not asked for this password.

3. **If others are using the system, type a warning message in the next field.**

   Your message is broadcast to other users.

   If you are the sole user of this system, you are not asked for this information.

4. **Type a number specifying the delay, in minutes, between the time the message is transmitted and the time the other users must have completed logging out.**

   Give the other users a reasonable amount of time to save their work and to log out.

   If you are the sole user of this system, you are not asked for this information.

5. **Click Shut Down.**

If you want to shut down and start up again immediately, choose Restart, instead of Shut Down, from the Special menu. The Restart dialog box appears. The procedure is the same as the shutdown procedure.

Shutting down from an A/UX CommandShell window

Whether you choose the Shut Down menu item, or enter `shutdown` from the A/UX command line, the `shutdown` program

- stops all daemons and kills all remaining processes
- executes `sync` a number of times to flush the write buffers
- unmounts the file systems
- executes `sync` again

The `shutdown` program entered at the command line performs this additional action:

- executes the `init s` command to bring the system to single-user mode

If you are running the console emulator, you must enter `shutdown` at the command line (as the root user). This action brings the system into single-user mode. After the `shutdown` program is completed, you can either reboot or power down the system. Follow these steps to shut down from the command line:

1. **Log in as the root user. Then enter**

   ```
   shutdown
   ```

   The system responds by printing a banner and the date

   ```
   SHUTDOWN PROGRAM
   Wed Jan 17 03:04:45 1990
   ```

   followed by a shutdown delay—the interval that the system will wait before beginning the shutdown procedure.

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

2. **If the default of a two-minute delay is acceptable, press RETURN. If you need more time, for example, if processes are still running or you need to notify other users, or less time because there are no background processes running and no other users, enter**

   ```
   y
   ```

The following prompt is displayed:

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

Enter `0` to start an immediate shutdown. To delay the shutdown, enter the number of minutes to elapse before shutdown begins. When you delay the shutdown, the following prompt about sending a special message is displayed:

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

3. **To enter your own message, enter `y` (yes) and you will be prompted.**

   Enter whatever text you want to send and then an end-of-file (usually CONTROL-D).

4. **To print the default message, enter `n` (no). 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.
```

5. **One minute later all users are informed that the system is going down in 60 seconds.**

   Then a message is displayed that asks:

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

   Enter `y` to put the system in single-user mode. When the `root` prompt is displayed, you have four options:

   - Enter `init 2` for multi-user mode, which returns you to the Login dialog box
   - Restart the system
   - Power down
   - Work in single-user mode in the console emulator.

6. **To restart, enter**

```
reboot
```

   which is the same as pressing the POWER ON switch from the power down state.

7. **To power down, enter**

```
powerdown
```

   which automatically turns off the power to the CPU and console. Then turn off the power to any external hard disks, in any order.

A/UX Startup program

A/UX Startup is a Macintosh Operating System program that passes control of the machine from the Macintosh OS to A/UX. It resides in a small Macintosh partition called MacPartition on a disk that contains A/UX. A **partition** is a part of a disk set aside for a specific use. Its value is in separating subsets of information for easier management. For a Macintosh file system and the A/UX operating systems to reside on the same disk, they must occupy separate partitions.

The A/UX Startup program is called a stand-alone shell because it is very similar to the A/UX shells. It is a command interpreter and has a command language with shell variables, comments, input and output redirection, and built-in commands. The programs that run from A/UX Startup can be read only from a Macintosh file system.

During installation, A/UX Startup can be set as the default startup application, so that the system always starts up in A/UX Startup. (See "Changing the Startup Device and Application," later in this chapter.) The pull-down menus above the A/UX Startup window provide Macintosh system-style interaction with the machine. Simultaneously, the command-line prompt in the A/UX Startup window allows you to use a subset of A/UX commands.

For additional information on this program, see `StartupShell`(8) in *A/UX System Administrator's Reference.*

A/UX Startup window

After you enter A/UX Startup, you see the startup shell window with its shell prompt. You can change the prompt to another string by redefining the shell variable `PS1`. For example, if you want the A/UX Startup prompt to be `hello:`, enter

```
PS1='hello: '
```

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 Sash — Displays introductory information about A/UX Startup.

Help — Displays the default help messages in the A/UX Startup window. Enough information is provided to enable you to use the `help` command.

Desk Accessories — Provides a list of the desk accessories currently installed on the system. You can invoke these using the mouse.

File menu

Close — Closes the currently active window. The A/UX Window, however, cannot be closed.

Quit — Exits A/UX Startup. It has the same effect as the `exit` command.

Edit menu

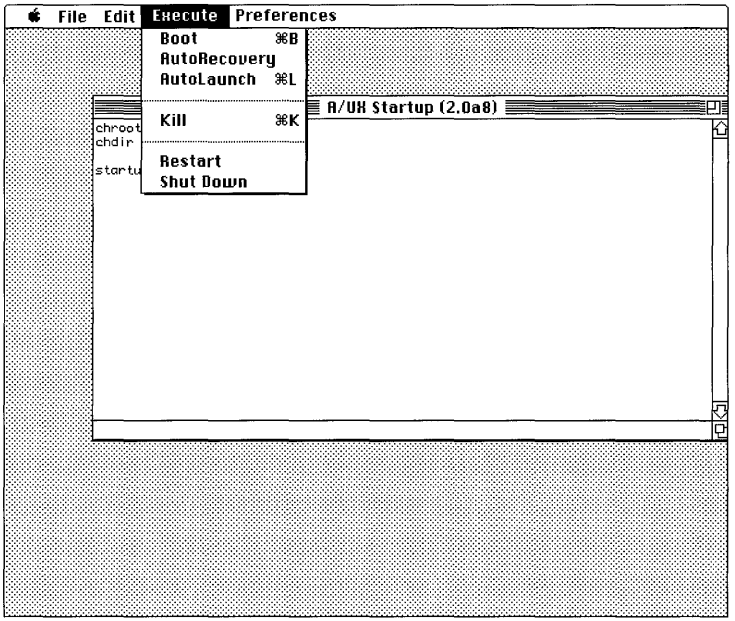
Undo, Cut, Copy, Paste, and Clear
Except for Copy, these items are to be used only with desk accessories. You can use Copy to copy selected text in the A/UX Startup window.

Execute menu

Figure 2-6 shows the Execute menu.

Boot — Performs the `autorecovery` and `autolaunch` commands. It is the same as the `boot` command.

■ **Figure 2-6** A/UX Startup Execute menu

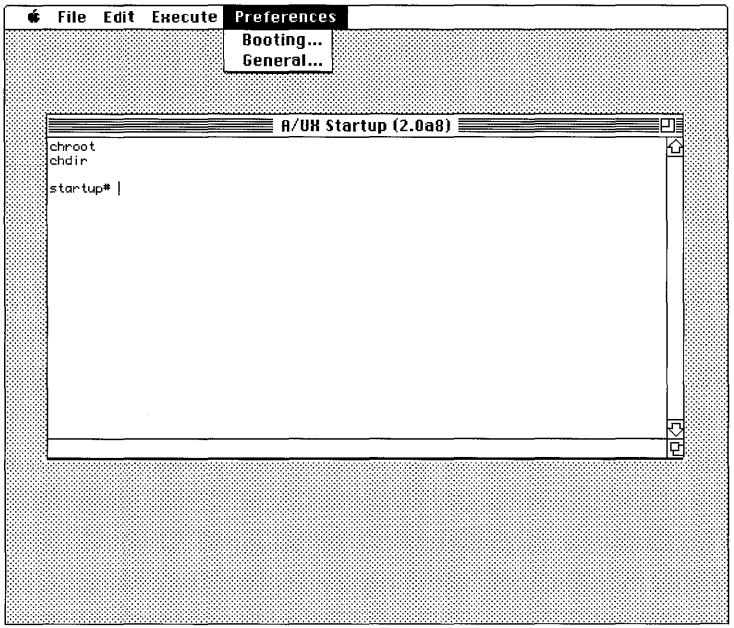


AutoRecovery	Performs `fsck` on the root file system, unless you enter another command in the AutoRecovery field in the Booting dialog box.
AutoLaunch	Performs the command options you assign to `autolaunch`. It is the same as the `autolaunch` command with no arguments.
Kill	Stops the currently running program. COMMAND-period and COMMAND-K are keyboard shortcuts for this item.
Restart	Restarts the machine. It is the same as the `restart` command.
Shut Down	Turns off the machine.

Preferences menu

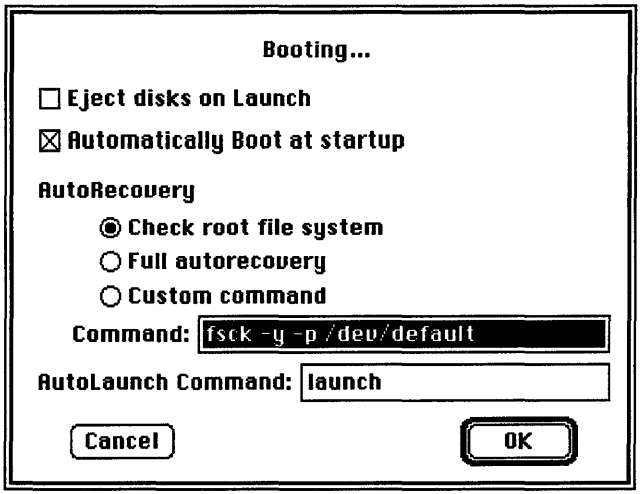
Figure 2-7 shows the Preferences menu

■ **Figure 2-7** A/UX Startup Preferences menu



Booting — Provides a dialog box that allows you to set startup parameters associated with the `boot` command, and the Boot item in the Execute menu (see Figure 2-8).

■ **Figure 2-8** Booting dialog box



Fields in this dialog box are described as follows:

Eject disks on Launch

When you select this box, all floppy disks are automatically ejected when the A/UX kernel is launched.

Automatically Boot at startup

When you select this box, A/UX automatically runs the boot command when launched, causing A/UX to boot as a part of the launching of A/UX Startup.

AutoRecovery Command

This field displays the `fsck` command line that is automatically run in the Macintosh OS before the kernel is launched. You can change the value by selecting a different radio button, or by selecting this box and editing the text.

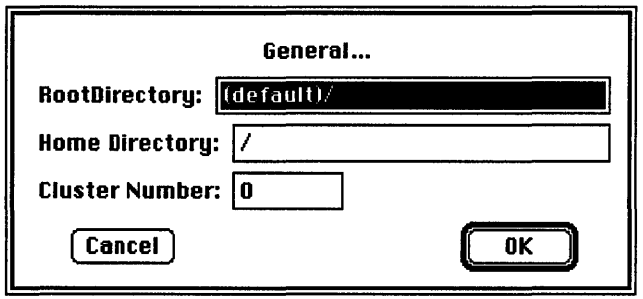
AutoLaunch Command

This text window displays the value of the built-in `autolaunch` variable. You can change the value by selecting this box and editing the text. To display console messages during the boot sequence, enter `launch -v` in this field.

General

The General menu item, as shown in Figure 2-9, provides a dialog box that contains the following miscellaneous items you may want to change.

■ **Figure 2-9** General dialog box



RootDirectory

This text window displays the value of the built-in `root` variable. To change it, select this box and edit the text. To boot A/UX from the same hard disk as A/UX Startup, use *(default)/*. To boot A/UX from a different hard disk, use *(n,0,0)/*, where *n* is the SCSI ID of the hard disk that contains A/UX.

Home Directory

This field displays the value of the built-in `home` variable. To change the value, select this box and edit the text.

Cluster Number

This field displays the value of the `autorecovery` cluster number. See `autorecovery`(8) in *A/UX System Administrator's Reference* for an explanation of this number.

Commands that run in A/UX Startup

The list of the commands you can run in A/UX Startup follows. These are not the actual A/UX commands, but rather functionally identical, stand-alone versions of the A/UX commands, written for use in A/UX Startup.

```
read_disk
cat             chgrp           chmod           chown
cp              cpio            date            dd
dp              ed              esch            fsck
kconfig         launch          ln              ls
mkdir           mkfs            mknod           mv
od              newfs           pname           rm
stty            svfs:fsdb       tar
ufs:fsdb
```

- *Note:* The `launch`, `esch`, and `read_disk` commands have no A/UX counterparts.

Because these commands enable you to work on the A/UX file system from outside A/UX, they are especially useful for troubleshooting A/UX. For example, you might edit `/etc/inittab` without running A/UX if you suspect that that file is causing you problems at boot time. You can list the contents of the root directory by entering

```
ls -CF /
```

Similarly, the command

```
ls -CF /users/start
```

lists the contents of the `/users/start` directory.

For short explanations of these commands, use the A/UX Startup help facility, which is described in "A/UX Startup Menus," earlier in this chapter. Since for all practical purposes these commands are identical to the A/UX commands of the same name, refer to the command descriptions, or man pages, in *A/UX System Administrator's Reference* or *A/UX Command Reference* for more complete information.

When `autoconfig` automatically reboots the system

A/UX automatically reboots when `autoconfig` finds an inconsistency between the cards in the expansion slots and the device drivers in the kernel and determines that the system needs a new kernel to correct the problem.

The system runs `autoconfig` before entering multi-user mode. The `autoconfig` program checks the consistency between the hardware in the expansion slots and the information contained in the kernel regarding software slot device drivers. The `autoconfig` program ensures that the information between the two is consistent. If so, it exits without rebuilding the kernel.

If the kernel contains a device driver and the matching expansion card is not present, `autoconfig` rebuilds a new kernel without that device driver and reboots the new kernel. For example, if you no longer have an EtherTalk™ card in your system, `autoconfig` detects the change, removes the associated device driver from the kernel, and reboots the system with the new kernel. This action protects you from performing input and output operations to a nonexistent card. It also means that if you add a device driver for an expansion card but then forget to add the actual card, `autoconfig` removes the device driver when the system boots. When you later add the matching card, you will need to build a new kernel using the `newconfig` command while running A/UX.

If the kernel contains a slot device driver and the matching card is present in a different slot than the kernel expected, `autoconfig` makes the appropriate changes to the kernel and reboots. This lets you install a card in any slot; `autoconfig` automatically makes the appropriate changes to the kernel.

- *Note:* If the system contains a slot card for which the kernel does not contain a driver, `autoconfig` does not build a new kernel. In this case, `autoconfig` displays a warning message, which is hidden by the boot sequence, and continues. This action allows you to add the appropriate software driver at a later time. To display the console messages during the boot sequence, pull down the Apple menu and choose CommandShell.

Changing the startup device and application

If you have more than one hard disk, you may have the option of configuring any one of them to boot the system at startup time. The startup disk must contain a Macintosh partition with a System Folder. If an A/UX disk is to be the startup device, it should contain A/UX Startup and a Macintosh System Folder. For instance, you might want to change the startup device if one hard disk is devoted to the Macintosh OS and another is devoted to A/UX.

You can also set which application the system automatically runs upon system startup. Any Macintosh application can be the startup application, including A/UX Startup. As shipped, the system boots up in the Macintosh Finder. This enables you to interact with certain Macintosh files when you begin using the system; for example, the README file. If you plan to work mainly with A/UX, you will probably want to set A/UX Startup as the startup application.

The capability of booting automatically into the application of choice is provided for convenience. You can simply press the POWER ON key to have your favorite application open immediately after startup. Instructions for setting the startup application in the Macintosh and A/UX operating systems are given in *A/UX Essentials.*

Changing the startup device

Initially, A/UX is configured to boot from the disk that contains A/UX Startup. You can change this to have A/UX boot from another device, from SCSI ID 0 through 6. Note that A/UX does not have to be on the disk that is the startup device. The procedure to change the startup device involves selecting the correct device from the Control Panel desk accessory to let the system know where to start.

1. **Turn on the computer and enter the Macintosh Operating System.**

   Cancel the A/UX boot, if necessary.

2. **Open the Control Panel desk accessory.**

   You choose the desk accessories from the Apple menu at the left side of the menu bar.

3. **Click the Startup Device icon and select a startup device.**

   You have to scroll down to see the Startup Device icon. Icons representing all possible startup devices appear on the right of the Control Panel display. Select one and close the Control Panel.

4. **Choose General from the A/UX Startup Preferences menu.**

   If the Preferences menu is not displayed, double-click on the A/UX Startup icon and then cancel the automatic startup screen. The dialog box shown earlier in Figure 2-9 appears.

   The Root Directory box contains the *(default)/* parameter, which is the disk on which A/UX Startup resides. Enter `default` at the command line to find out which SCSI ID *(default)/* refers to.

5. **If A/UX Startup is not on the disk that contains A/UX, replace the *(default/)* parameter with the SCSI ID number of the device you want to use as the startup device: *(ID,0,0)/*.**

   Normally the SCSI ID number assigned to an external hard disk is 5, unless you reassigned it when preparing it for A/UX. You can verify the SCSI ID number by checking the indicator on the back of the external hard disk.

6. **Click the OK box.**

7. **Reboot the machine by choosing Restart from the Execute menu.**

   Your system will now boot from the disk configured as the startup device.

Making A/UX Startup the startup application in the Mac OS

The startup application is the program that appears on the screen after you turn on the computer. The ability of the user to set a startup application is a feature of the Macintosh OS, so you choose a startup application from the Macintosh Finder. You can make any application your startup application. For example, if you use A/UX Startup in most of your sessions with the computer, it makes sense to set A/UX Startup as the startup application. See your Macintosh documentation or *A/UX Essentials* for instructions.

Technical details

This section describes the steps the Macintosh follows when looking for startup files and the technical details of the procedures used to boot A/UX from a disk.

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
3. The hard disk with the highest SCSI ID number

- *Note:* The standard Apple® Hard Disk 20 is connected to the system in a way that is similar to a floppy disk drive. If a bootable Hard Disk 20 is connected, it will be the startup disk.

You can override the natural order by setting the hard disk you want to start from as the startup device.

How A/UX boots from a hard disk

A/UX is designed to be booted from any SCSI disk, from SCSI ID 0 through SCSI ID 6. The A/UX kernel accepts boot device information from A/UX Startup. This design is unlike many other UNIX kernels, which frequently have the boot device hardwired into the kernel and require use of `adb` or similar programs to patch in boot device changes.

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. The A/UX kernel does not recognize disks in general; they become part of the kernel. 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 (from one of several possible places) 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.

Customizing your system

You can set the following for your system:

- the time zone
- the name of your system
- the message of the day
- the mode your system automatically boots into (multi- or single-user mode)

◆ *Note:* The mouse tracking speed, which is controlled by the kernel, is the same for all A/UX users. It cannot be changed.

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. The Macintosh Operating System does not change for daylight saving time. Because the A/UX clock can adjust for these variations, run the `date` command to change the time whenever it is affected by daylight saving time.

△ **Important** The Macintosh OS time should never be reset because it is automatically synchronized to the A/UX time. △

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. Changing the time through the Control Panel does *not* affect A/UX time. A/UX uses the time and date to monitor or control a number of activities. Without the correct time, the A/UX system cannot properly

- 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

When you enter the `date` command, A/UX calculates your local time by using the GMT bias. The system clock is set at the factory to Pacific standard time. If you live in another time zone, set the local time zone when you first set up your system. Follow these steps, depending on whether you use the A/UX Finder or the command line.

- **From the Macintosh interface:**
  - Log in as the root user.
  - Open Useful Commands in the Useful Commands folder.
  - Double-click on the Settimezone icon.
- **From the A/UX CommandShell or console emulator:**
  - Become the root user.
  - Enter `settimezone` from the A/UX command line.

After you have completed this procedure, a list of the major time zones by region is displayed (see Figure 2-10).

■ **Figure 2-10** Time zone menu

```
/Useful Commands/settimezone (set timezone)
Enter the letter corresponding to your region.
        f  Africa
        a  Australia, New Zealand
        c  Canada
        e  Europe
        i  Iceland, Caribbean
        x  Mexico
        m  Middle East
        s  South America
        u  United States
        w  Western Pacific, East Asia
        o  Other

        -  Cancel
Enter letter: w
```

Enter the letter that corresponds to your region. For example, if you live in Japan, enter `w` for Western Pacific. A second menu is displayed that lists time zones with the region (see Figure 2-11). Enter `j` for Japan. If you live in the United States, enter `u` as the region. In the second menu that is displayed, enter the letter that corresponds to your United States time zone. For example, if you live in New York, enter `e` for Eastern standard time.

■ **Figure 2-11** Time zone submenu

```
Enter the letter corresponding to your time zone.
        c  Peoples Republic of China
        h  Hong Kong
        j  Japan
        r  Republic of China
        k  Republic of Korea
        m  Samoa
        s  Singapore

        -  Return to previous menu
Enter letter: j

The time zone has been set to Japan
The corresponding date will be Mon Feb 19 23:08:21 JST 1990
This change will take effect the next time you log in.
#
```

If your time zone is not listed, select the Other option, which enables you to select your time zone relative to Greenwich Mean time. You will enter the difference between your time zone and that of Greenwich, England. To help determine this offset, refer to Figure 2-12, which shows a world map divided into time zones one hour later than (east) or one hour earlier than (west) of Greenwich.

Locate your region and enter the letter in the menu associated with the correct number of hours that your time zone is greater or less than GMT time. For example, according to the map, New Delhi is located five time zones east of Greenwich. Regions east of Greenwich are measured in one-hour increments greater than the GMT, while those west of Greenwich are measured in one-hour increments less than the GMT. Therefore, for a system located in New Delhi, you would enter `f` for five hours later.

If you enter the GMT offset and your time is currently affected by daylight saving time, you should add one hour. So if New Delhi is currently on daylight saving time, enter `g`, for six hours later instead of five.

After you enter the time zone by either method, a message is displayed that says the time zone has been set to the selected country or region, or for GMT offset—the number plus or minus the GMT—for example, GMT+5. The corresponding date and time are also given. This change takes effect the next time you log in.

If `settimezone` fails to set the A/UX time correctly, the hardware clock is incorrect. Use the `date` command to reset it.

Resetting after moving a system to a different time

If you move the computer to a different time zone, adjust the GMT bias using the `settimezone` command described in the previous section.

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 command you enter, however, depends on your command shell. For example, a user logging in from Oregon and working in the C shell, which normally sets environment variables from the user's `.login` file, would enter

```
setenv TZ PST8PDT
```

A user logging in from California and working in the Bourne or Korn shell, which normally sets environment variables from the user's `.profile` file enters

```
TZ=PST8PDT
```

Changing the message of the day

If you log in to the Bourne or Korn shell, the `/etc/profile` system startup script executes several commands, including

```
cat /etc/motd
```

If you log in to the C Shell, `/etc/cshrc` executes this command.

In response, the system displays a message contained in the file `/etc/motd` (for "message of the day"). You can change the contents of `/etc/motd` to anything you like by editing the file with a text editor. The new text you enter will be displayed the next time you bring the system down to single-user mode, reboot, or log in.

Renaming the system

To change the system's name, use a text editor to open the `/etc/HOSTNAME` file. The default system name is `localhost`. 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. Change the first field and save the file. This change takes effect the next time you shut the system down or reboot.

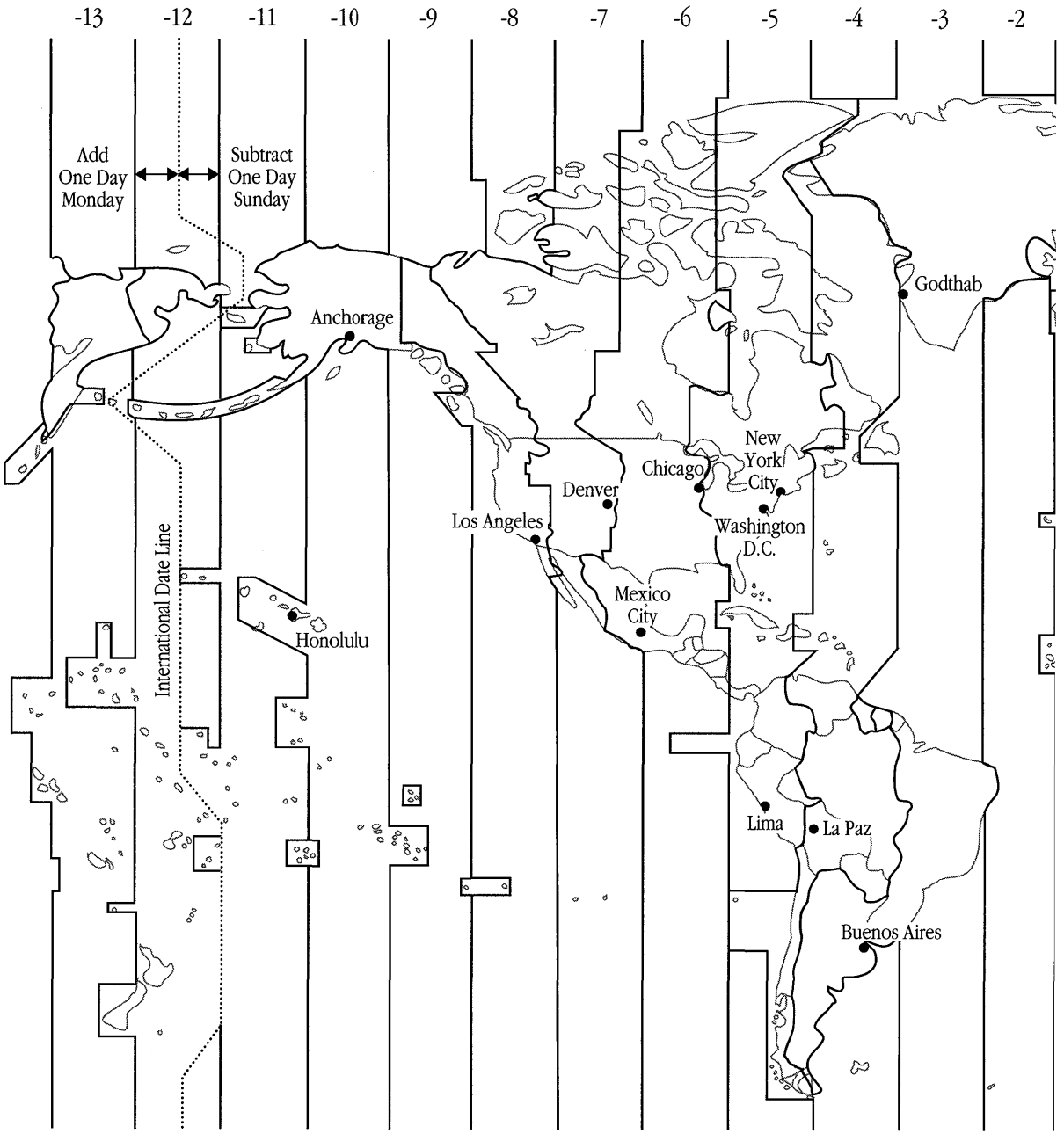
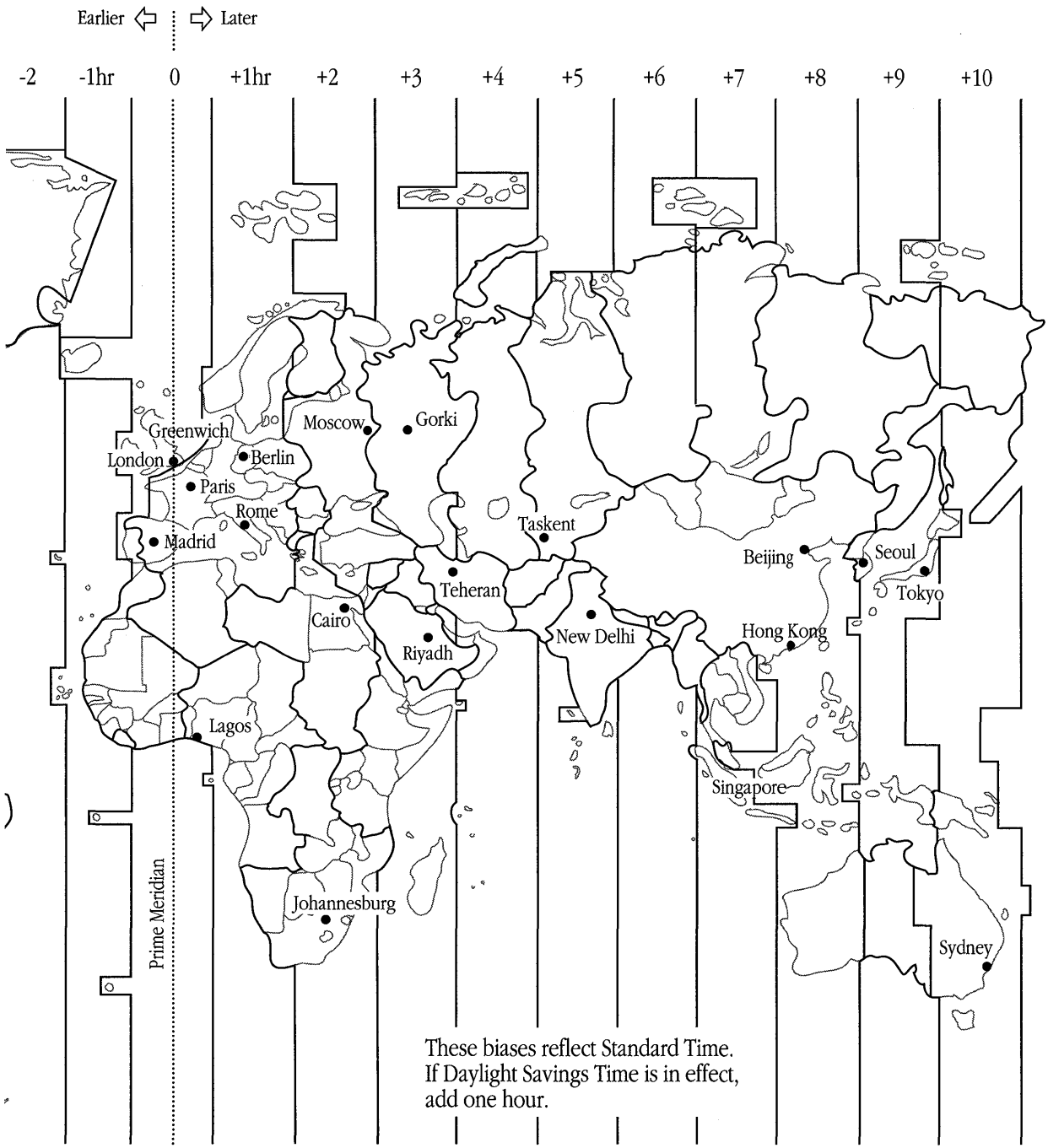


Figure 2-12 The GMT bias map


Earlier
Later
-2
-1hr
0
+1hr
+2
+3
+4
+5
+6
+7
+8
+9
+10
Greenwich
London
Berlin
Paris
Rome
Madrid
Moscow
Gorki
Taskent
Teheran
Cairo
Riyadh
New Delhi
Beijing
Seoul
Tokyo
Hong Kong
Singapore
Lagos
Johannesburg
Sydney
Prime Meridian
These biases reflect Standard Time.
If Daylight Savings Time is in effect,
add one hour.

Single- and multi-user modes

The system automatically starts up in multi-user mode, the preferred mode for most activities. For the initial system administration duties that require single-user mode, see the description of the `shutdown` program in "Shutting Down from an A/UX CommandShell Window," earlier in this chapter, which describes how to enter single-user mode.

If your system has one or just a few users, the two basic run levels—single-user and multi-user—will probably suffice for normal operation. Descriptions of these two modes follow. If you have a system with many users and perhaps many modems, you may be interested in more complex run-level capabilities, described in "Changing Run Levels: `init`," later in this chapter.

Single-user mode

Single-user mode is one of several **run levels** available on the system. As the name implies, in single-user mode, only one person has access to the computer because the terminal ports are disabled. Also, the `init` program does not launch the background processes specified in `/etc/inittab`. This quiet state of the machine is necessary for the performance of administrative tasks, such as

- Checking the root file system with the `fsck` command (although it is better to check it from A/UX Startup), and
- Copying the file systems to backup media.

In single-user mode, you have all the privileges of root. You can create or destroy anything on the system—files, directories, or processes—in just a few keystrokes. This level of power is necessary to perform most administrative tasks, but you need to use it selectively. Unless you need the power and privilege of single-user mode, run A/UX in multi-user mode—the default. This good habit decreases the potential of your making a mistake that could damage the system. It also deters system abusers from sneaking in and wreaking havoc on the system.

To boot A/UX into single-user mode automatically each time the system is rebooted, edit the file `/etc/inittab`. Change the line

```
is:2:initdefault
```

to

```
is:s:initdefault
```

In single-user mode, the system uses a console emulator interface, not the A/UX Finder interface.

- *Note:* You can edit the `/etc/inittab` file from A/UX Startup, if necessary, before booting A/UX.

Multi-user mode

The general 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 daemons are enabled, as well as all ports designated in the `/etc/inittab` file to permit user logins.

To enter multi-user mode from single-user mode, enter

```
init 2
```

You will now see

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

As the system enters multi-user mode, commands stored in `/etc/inittab`, `/etc/rc`, `/etc/brc`, and `/etc/bcheckrc` are executed (see "Initial Processes: `/etc/inittab`," later in this chapter). When this process is complete, you should see the Login dialog box. The system is now running in multi-user mode, and you may log in.

Changing kernel parameters

There are two reasons to change kernel parameters. The first is to take advantage of your system's abundance of main memory by increasing the disk buffer cache. This is accomplished by increasing the NBUF parameter. NBUF is set to automatically configure based on 10 percent of the available memory. To increase this amount, you have to specify a value. The second reason is a need to exceed the kernel-configured limits on your system. Such limits are evident from the error messages that appear. It is best not to change kernel parameters (except NBUF to increase the disk buffer cache), unless you see one of these screen messages:

Error message	Parameter requiring an increase
`inode: table is full`	`NINODE`
`file: file table is full`	`NFILE`
`proc: table is full`	`NPROC`

If you encounter one of these messages, enter the following commands (you can increase one or more of the parameters) to reset these kernel parameters:

```
kconfig -n /unix
NINODE=400
NFILE=400
NPROC=200
```

Press CONTROL-D.

It is safe to further increase these parameters (`NINODE`, `NPROC`, and `NFILE`) each time you run into kernel-configured limits.

You must reboot to put the new parameters into effect because `kconfig` changes the kernel file and not the currently running kernel. See `kconfig`(1M) in *A/UX System Administrator's Reference* for an explanation of these parameters.

Initial processes: `/etc/inittab`

An overview of the initial processes follows. For more complete information, refer to the `init`(1) man page. The system's topmost program is `init`, initial process. It is the first process to run after you boot the system. The `init` program spawns processes as specified in the `/etc/inittab` file.

One of the first commands `init` executes is the `/etc/sysinitrc` shell program, which performs such basic functions as setting the system's clock.

The next line in `/etc/inittab` specifies the default initial run level:

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

The run level in the `/etc/inittab` file is specified in the second field of each entry, in this case `2` for "multi-user." Once the initial run level is determined, `init` processes only those `/etc/inittab` entries whose *run-level* field is the same as the run level currently in effect.

If you check the contents of `/etc/inittab` on your system, you will see the processes invoked by `init`. For example, the following processes started by `init` are found in `/etc/inittab`. (The process names are preceded by an *id* and a *run-level*, as discussed in the next section.)

`/etc/bcheckrc` A startup script that runs `fsck` on those file systems other than root.

`/etc/brc` A startup script that sets the permissions on pseudo-`ttys`.

`/etc/rc` A startup script that mounts the file systems (if applicable) and performs some general housekeeping.

`/etc/getty` A process that enables logins on serial ports. See Chapter 3, "User and Group Administration," and Chapter 7, "Managing Other Peripheral Devices."

`/etc/inittab` entry format

Each line in `/etc/inittab` is an entry containing four fields separated by colons and followed by an optional number sign (#) and an optional comment. The format of each entry is *id:run-level:action:command*

where

id Unique entry identification

run-level Run level at which the entry is processed

action Action to be taken with next field

command Command to be executed

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
```

Comments about the line are preceded by a number sign (#). These comments indicate that the line refers to the console port. Because the *action* field is set to `respawn`, this port is enabled in multi-user mode. This is explained in more detail in the discussion of the *action* field that follows.

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 for an entry that affects a port be the corresponding port number. In the *id* field of the above line, `co` refers to the console port.

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 the system's run level.

`respawn` Tells the system to run this command when the designated run level is entered. If the command terminates for any reason, the system should run it again. If you specify `respawn` as the *action* field, the command runs whenever the run level matches the run level of the entry. If you leave the *run-level* field empty, the process runs at all times and all run levels.

`off` Tells the system to turn off this command for all run levels.

`once` Tells the system to turn on this command for this run level. If the command terminates for any reason, no further action is taken.

`wait` Tells the system to wait until the current command is completed before `init` goes to the next line. The `wait` 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 each entry matches the default run level, the command named in the fourth field is executed. In the above example, the command `/etc/loginrc` would be executed for each entry in the `/etc/inittab` file whose command field is `/etc/loginrc` and whose *action* field is `respawn`, as modified by the *action* field.

Changing run levels: `init`

Once the system is started and running at the default run level, you can change the run level. Log in to the console emulator as the superuser and enter

```
init run-level
```

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

When you enter this command, `init` scans `/etc/inittab` again, kills processes that should not be active in the new run level, except for those that spawn daemons (see the discussion of `init Q` 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.

For instance, if the following lines appear in `/etc/inittab`

```
01:23:respawn:/etc/getty tty1 at_9600
02:2:respawn:/etc/getty tty2 at_9600
```

both ports `tty1` and `tty2` are running a `getty` command when the system is at run level 2. But if you later type

```
init 3
```

the `getty` running to `tty2` is terminated, whereas the one running to `tty1` continues because both run level 2 and run level 3 are specified for that port.

The only way to modify how a process runs at a new run level is to specify it in `/etc/inittab`. In theory, you should specify every process that affects, for instance, a port, with reference to every single possible run level. In practice, you do that only for those processes that you may want to turn on and off for certain specific run levels. See Chapter 7, "Managing Other Peripheral Devices," for examples.

If you introduce a change in `/etc/inittab`, enter

```
init Q
```

to make the change known to the system without restarting it. This command forces `init` to reread `/etc/inittab`. Some of the processes controlled by `init` spawn child processes, however, and these child processes are not terminated by the `init Q` command. You can terminate these processes individually by specifying their process ID number to the `kill` command, or by shutting down the system and rebooting.

The general functioning of your system is determined by the current run level. The current run level is determined at startup by the default entry in `/etc/inittab`, and it can be changed via the `init` command. The current run level determines which commands (in the command field of `/etc/inittab` entries) are activated.

When running `init`, you should be careful not to bring about any changes that would disrupt the activities of users currently logged in. For instance, `init` could disrupt an operating modem if a `getty` command were sent to that port. Know your run states and make sure that changing them will not disrupt user activity.

Screen locks, power failures, and emergencies

If your system freezes and you are unable to invoke a command, you have several options. For example, if you have modified `/etc/inittab` to start a `getty` on another terminal, you can try to log in as the root user from another terminal. Or, 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. Pressing COMMAND-CONTROL-E terminates all active Macintosh applications, if possible. Sometimes simply waiting a few minutes unlocks the screen.

If these measures fail, or if an emergency occurs, bring down the system by pressing the power switch at the rear of the unit, or by using the programmer's switch. These actions should be your last resort because they can introduce inconsistencies in the file system. In a normal shutdown, file systems are unmounted before the power is turned off. Of course, in an emergency or if the power goes off, an ungraceful shutdown is your only choice.

When you reboot, the file system check program, `fsck`, is run automatically. If you receive the message:

```
The file system at /mnt needs repairs. Repair automatically?
```

in the dialog box, click Repair. If the message reappears, bring up CommandShell (choose it from the Apple menu) to see exactly where `fsck` is encountering problems. Then check the file system by interacting with fsck through the A/UX Startup program as described in Chapter 8, "Checking the File System: `fsck`."

In this book, troubleshooting tips for a procedure usually follow the discussion of the subject. For overall troubleshooting, see Chapter 11, "Troubleshooting."

Chapter 3 **User and Group Administration**

This chapter discusses the user's working environment and group administration. The major topics covered include

- The user's working environment
- The administrator's role in assigning permissions
- File permissions
- Adding users to the system
- Modifying the user's working environment
- Removing users
- Troubleshooting

This chapter assumes that you are familiar with the related material in *A/UX Essentials.*

The user's working environment

Within the A/UX system, each user has a working environment that is analogous to a worker's personal space. Each user's environment 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, 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 a mechanism 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:** In most cases, 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.

This chapter describes procedures and concepts that will assist you in establishing, modifying, and removing users' working environments.

Components of a user's 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."

The following are the components of a user's working environment:

- **Permissions:** Every file in the A/UX system is associated with **permissions**, also known as **modes**, which determine who can do what with the file. Three kinds of actions can be performed on files, each associated with a corresponding type of permission: write, read, and execute. These three types of permission can be set differently for three types of users: the owner of the file, users belonging to the same group as the owner of the file, and all other users. Users can execute programs or have access to data files only if permissions are set appropriately for each program and data file. (See "Permissions," later in this chapter.)

  Macintosh folders, but not files, have read, write, and execute access permissions. For a complete discussion of Macintosh folder permissions, see "Protecting Your Files and Folders," in Chapter 2 of *A/UX Essentials.*

- **Login name and password:** Before gaining access to the system, a user must enter his or her login name and password, which are defined in the `/etc/passwd` file. (See "Files That Determine a User's Environment," later in this chapter.) Note that multiple users can share one account. In this case, the login name is the account name, and the password is the account's password.

- **User ID:** Each user login name is associated with a unique numeric user ID (**UID**) that identifies the user. This is defined in the `/etc/passwd` file, which is also described in "Files That Determine a User's Environment," later in this chapter. When a user attempts to use a command on some data, this number is compared to the user ID associated with both the command and the data file. If there is a match in either case, the files are checked to see how the permissions are set for the owner of the file. If there is no match, the user's group ID is compared to the group ID associated with the files. If it also fails to match, the permissions for *other* users apply.

- **Group ID:** Each user login name is associated with at least one group ID—**GID.** This is defined in the `/etc/passwd` file; and also described in "Files That Determine a User's Environment," later in this chapter. The group ID numbers indicate the groups to which the user belongs at the time of login. Group membership is a security feature that permits some users access to files, while denying this access to other users. This inclusion and exclusion can be total or partial. Group names and additional group IDs are defined in `/etc/group`.

- **Home directory:** Each login name is associated with a **home directory** usually located in the `/users` directory, which is defined in the `/etc/passwd` file; and also described in "Files That Determine a User's Environment," later in this chapter. When users log in to the system, their default shell programs use this home directory as the home base for creating and using files. For example, in A/UX the conventional home directory associated with the login name `paul` should be `/users/paul`, not `/usr/paul`. (The default is `/users/`*login-name*.) The user can modify special files residing in this directory to tailor his or her environment further.

  You can access your home directory either through the `/users` directory from the A/UX command line, or from the /users Macintosh folder. See the following section, "Macintosh Personal and System Folder Considerations."

- **Current directory:** This is the directory in which the user is located at the present time. 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:** These are files that are executed each time a user logs in; their contents determine the user's environment.

- **Default shell program:** After logging in, the user needs some way of communicating with the system. The **default shell program** is the program that automatically greets the user after a successful login. This is defined in the `/etc/passwd` file and is also described in "Files That Determine a User's Environment," later in this chapter. The default shell program is usually defined as the C Shell, Bourne shell, or Korn shell. You can use any of these three A/UX command interpreters as the default shell program. If the field is empty, the default shell program is the Bourne shell. The default shell for the `adduser` command, which enables you to add users quickly and is discussed later in this chapter, is the C Shell. Sometimes the default shell program is a more restricted and restrictive program than the A/UX shells. See "Changing a User's Default Shell Program," later in this chapter, for more information.

- **Other environment variables:** The user's shell provides a number of environment variables that can be assigned different values to alter the environment. 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` (default shell program). Other variables are assigned values at the shell prompt or in files such as `.login` or `.profile` in each user's home directory. One such variable that affects the user's working environment is the `PATH` variable. The value assigned to a user's `PATH` variable determines which directories, and in what order, the shell will search for the file corresponding to a command issued by the user. For other environment variables, see *A/UX User Interface.*

Macintosh personal and System Folder considerations

Whenever a directory is created under A/UX using the `mkdir` command, the directory is represented as a folder icon on the A/UX Finder desktop. This dual representation of the same directory enables users to navigate the A/UX file system using the Finder, as well as the A/UX command line. See "Getting Started with A/UX" in Chapter 2 of *A/UX Essentials* for a complete discussion of this topic.

If necessary, users can create their own System Folders within their home directory folders by using the `systemfolder` program. For example, a user may want to have a personal System Folder (instead of using the shared one in `/mac/sys`) to display a clock in the menu bar. Using the global System Folder is preferable, however, because each local System Folder consumes half of a megabyte of memory at creation and increases in size with use.

Files that determine a user's environment

The user's environment is normally established at login time from information stored in a number of system files. See "How A/UX Establishes the Environment," later in this chapter, for a description of the order in which this is done. This section describes each of these files, along with its format and content.

The /etc/passwd file

The user's working environment is specified largely by a single entry in the /etc/passwd file. You must be logged in as the root user to modify this file. To monitor the /etc/passwd file, use the password check command, pwck(1M), which generates a report that shows inconsistencies in the file.

The /etc/passwd file distributed with A/UX has several administrative logins and the user logins start and Guest. See "Administrative Logins on the A/UX System," in Chapter 1, for a description of the administrative logins.

◆ *Note:* Yellow Pages password information differs. See *A/UX Network System Administration* for details.

Each entry consists of one line with seven fields separated by colons. The form of an entry is

login-name:*password*:*UID*:*gid*:*misc*:*home-directory*:*startup-program*

where the fields are interpreted as follows:

login-name — The name the user must use when logging in. It must be unique in the /etc/passwd file. This name should be no longer than eight characters.

password — An encrypted version of the actual password the user must use when logging in. The encryption is done automatically when the password is first assigned and whenever it is changed. Having the password encrypted makes it possible to have the /etc/passwd file open for reading by everybody.

UID — A unique user ID number for each user.

GID — The user's default group ID. Even if the user is listed in the /etc/group file (see the next section, "The /etc/group File"), he or she belongs by default to the group whose number appears in the user's *GID* field in the /etc/passwd file.

misc Miscellaneous information about the user, such as full name, address, and telephone number. For other uses of the *miscellaneous* field, as well as for the exact specifications of the *password* field, see `passwd`(1) in *A/UX Command Reference* and `passwd`(4) in *A/UX Programmer's Reference.*

home-directory The name of an existing directory whose permissions, ownership, and group membership are such that the user can have access to it. Whenever the user logs in to the system, this is the directory in which he or she is initially located.

startup-program The name of an executable program, usually one of the A/UX shells, that permits the user to communicate with the A/UX system.

The following is a typical `/etc/passwd` entry:

```
joe:AxhlmzGfpRolE:101:100:Joe Doe:/users/joe:/bin/sh
```

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. To monitor this file, use the group check command, `grpch`(1M), which generates a report that shows inconsistencies in the file.

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

group-name:*password*:*gid*:*list*

where the fields are interpreted as follows:

group-name The group name. Group names are arbitrary, but by convention their meanings should be self-evident (for instance, `acctg` rather than `xyz24`).

password An encrypted version of the group's password. Although the *password* field can be set for each group, it is common practice to disable group password checking by filling the *password* field with the word `VOID` or with asterisks (`*`). In this case, a user can gain access only if listed in the `/etc/group` file.

GID A somewhat arbitrary number closely associated with the group name. For each group ID there is only one group name, and vice versa. The actual group ID numbers that exist in the `/etc/group` file are the only numbers that should be entered in the *GID* field of `/etc/passwd` entries. The group ID entered for each user in `/etc/passwd` should coincide with the group ownership of that user's home directory.

list A list of the login names of the members of the group. The login names must be separated by commas. Entering a user's login name in the group's list field is optional for those users who belong to only one group, but is recommended for accounting purposes. In this case, the group membership is determined by the value assigned in the *GID* field of their `/etc/passwd` entry.

The following is a typical `/etc/group` partial listing:

```
acctg:*****:100:paul,john,peter
legal:****:200:rose,paul,john,peter
```

By convention, group names are arbitrary although self-evident. The groups themselves, though, should not be arbitrary. As system administrator, you should divide users into groups according to their assigned activities, and allow users to belong to different groups only when there is some overlapping of activities. If a user belongs to too many groups, ask yourself whether the group distribution you have established is the best one.

As an example of the use of the `/etc/group` file to establish multiple group memberships for a user, look again at the preceding partial listing of an `/etc/group` file. You will notice that paul is listed in both `acctg` and `legal`. If the group ownership of his home directory is `acctg`, then every file he creates and every directory he makes below his home directory will also belong to `acctg`. For various reasons, Paul might want some of the files he creates to belong to the group `legal`. There are two ways he can accomplish this:

- He can create a file having group ownership `acctg` and then change its group membership to `legal` after the fact.
- He can make a directory having group ownership `acctg` and then change the group membership of the directory to `legal`; every file created within that directory will also belong to `legal`.

In both cases, Paul can change a file's or a directory's group membership by giving the command `chgrp`. For example, if Paul creates a file named `court` in his home directory (group membership `acctg`), a long listing of the file (`ls -l`) might show

```
drw-rw-r--  1 paul  acctg   512 Oct  3 17:51 court
```

To change the directory's group membership, Paul has to enter

```
chgrp legal court
```

After this command, a long listing would show the new group:

```
drw-rw-r--  1 paul  legal   512 Oct  3 17:51 court
```

Note that the file permissions are not changed by the `chgrp` command, but now they apply relative to the group to which the file belongs.

Now suppose that Paul wants to create a new file `court/notes` in his `court` directory. Before creating the file, the system checks to be sure that he has the appropriate permissions. First, his *UID* is checked. If his *UID* matches that of the parent directory, the file is created (assuming that he has write permission in the directory). This requirement would be met if Paul were trying to create the file `/users/paul/court/notes`. If his *UID* does not match the *UID* of the parent directory, the *GID* is checked. If the directory's group is a valid group for his account, the file can be created. Paul satisfies this condition when creating the file `/court/notes` because he is a member of the group `legal`. If neither of these conditions is met, the permissions of the parent directory are checked to see if all users are permitted to create files. If this condition is met, the file is created. The file is assigned Paul's *UID* and the *GID* of the parent directory.

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 at a time.) 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`. This causes *groupname* to replace the first group listed in your environment that is not in the `/etc/group` file 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.

Setup files

The setup files for the C Shell—`.chsrc`, `.login`, `.logout`—and for the Bourne and Korn shells—`/etc/profile` and `.profile`, and `.kshrc`, respectively, 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, which is discussed later in this chapter in "Adding a User Quickly: `adduser`," these files are copied to the new user's home directory.

- *Note:* When `$HOME` precedes a setup file name, such as `$HOME /.login`, it represents the user's home directory.

The `.cshrc`, `.login`, and `.logout` setup files

When a user logs in and the C Shell is specified as the setup file in `/etc/passwd`, the system automatically runs several shell scripts before giving the user a prompt. One of these scripts is the file `/etc/cshrc`. 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/cshrc` file, if it exists, is run *before* the file `.cshrc` in the user's home directory. A user can override any actions performed in the execution of `/etc/cshrc` by including the appropriate commands 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 not rerun. Therefore, you should place commands that need to be executed only once in `.login`.

When the user logs out, the commands in his or her `.logout` file are executed.

The /etc/profile and .profile setup files

When the Bourne shell or Korn shell is specified as the user's setup file in /etc/passwd, the system automatically runs several shell scripts before giving the user a prompt. One 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, if it exists, is run *before* the file profile in the user's home directory and thus serves as a default .profile. A user can override any actions performed in the execution of /etc/profile by including the appropriate commands in his or her own .profile.

The .kshrc setup file

When the Korn shell is specified as the file that is executed each time the user logs in, the system runs the /etc/profile and $HOME/.profile files described in the preceding section. (The setup file is part of the user's entry in the /etc/passwd file.) If one of these files sets the ENV variable to any filename ($HOME/.kshrc in the standard distribution), the system also reads the contents of the named file. After the initial login, whenever a system starts a new shell program, the file named in ENV is run again. (Note that /etc/profile and $HOME/.profile are not rerun.)

How A/UX establishes the environment

A close look at a successful login will help you understand how all the elements mentioned up to this point interact to determine a user's environment.

1. Upon a successful login, 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 command named as the user's default shell program in the *command* field of the /etc/passwd entry. This command 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`, `TERM`, and `EXINIT`; as well as set other features, such as aliases in the C Shell and functions in the Bourne shell. Try to keep modifications of this file to a minimum because they apply to all users of the system and are not easy for users to change. Edit the home files template found in `/usr/lib/skel` instead. See `sh`(1), `ksh`(1), and `csh`(1) in *A/UX Command Reference.*
3. After reading the default command file, the shell looks for and (if it exists) reads an initializing file in the directory named in the *home-directory* field of the `/etc/passwd` file. If the default shell program is the Bourne shell or the Korn shell, this file is called `.profile`; if it is the C Shell, there are actually two files, `.cshrc` and `.login`. In this file, the user can modify certain aspects of his or her working environment, such as `PATH`, `HOME`, `TERM`, and `EXINIT`; as well as set other features such as aliases in the C Shell or Korn shell, and functions in the Bourne or Korn shell.

   At this point, if the user is logging in to the A/UX Finder 32-bit mode environment, the login shell executes `/mac/bin/mac32` as its first and only command. If a `.mac32` file exists in the user's home directory, it is executed instead. For A/UX Finder 24-bit mode, these files are `/mac/bin/mac24` and `.mac24`, respectively.
4. Once the startup environment has been established, the default shell program prompts the user for input.

△ **Important** Avoid local modifications to the `/etc/cshrc` and `/etc/profile` files, which affect your users. Instead, use the standard files, such as `.cshrc` and `.login` found in `/usr/lib/skel`. These are the files used by the `adduser` program. △

The administrator's role in assigning permissions

The A/UX system administrator needs to provide adequate security for the users and projects on the machine. Initial security for accounts 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 should be an option, not the default. To ensure security, the administrator must set up appropriate entries in `/etc/passwd` and `/etc/group`.

Security on the Guest account is another issue. As shipped, the Guest account does not have a password. A feature of `Login`(1M) enables you to disable the Guest account by removing the *Guest* entry from the `/etc/passwd` file. If this entry exists and has not been modified, guests can log in without a password.

The system administrator can take the following steps to ensure security on the Guest account:

- **Set the password for Guest, which means that a user logged in as Guest can change it and not tell you (the default); or**
- **Use password aging to set the password so that only the system administrator can change it.**

Password aging is described in detail in the `passwd`(4) man page. In brief, only the superuser can change the password if you set the second character of the age field—the minimum period in weeks that must expire before the password can be changed—to be greater than the first character—the maximum number of weeks for which a password is valid. (The age field is entered after the encrypted password and is preceded by a comma.)

Passing the `-r` flag to `Login` in the `/etc/loginrc` file removes the System V password restrictions so that any password is acceptable. To do this, change the line to the following:

```
exec /mac/bin/Login -m2m -s '/mac/sys/Login System Folder' -- -r \
>/dev/console 2>&1
```

It may also be appropriate to set the `umask` in the system-wide shell initialization files.

When you set up common areas for group activity, it is often useful to create a user name for the project itself. Certain privileged users can then log in to that account and perform administrative tasks not allowed to all members of the project's group.

Permissions

Permissions determine who can and cannot have access to and use a particular folder, file, or directory. For example, if a file grants read-write permission to all users on the system, anyone on the system can get into the file and permanently change it. If a file grants read-only permission to all users on the system, all users can read the file, but no one except the owner can make changes to the file.

These permissions can be set for users in three classifications, called **access classes:** *user* (or *owner*), *group*, or *other*. Each access class is represented by three characters; the order of characters is `r`, `w`, `x`, indicating the access permission granted to that category of user. If a hyphen appears instead of an `r`, `w`, or `x`, permission to perform that action is denied to that category of user.

For the most part, this section addresses A/UX file and directory permissions. For a thorough discussion of Macintosh file and folder permissions, see Chapter 2, "Protecting Your Files and Folders," in *A/UX Essentials*.

File-access permissions

Users can set the following permissions on the files they own:

`r` Read permission. Allows designated users to read a file or to copy its contents.

`w` Write permission. Allows designated users to modify a file.

`x` Execute permission. Allows designated users to execute a file (that is, to run it as a command).

As shown in Figure 3-1, ten characters are used to represent a file and its permissions; the first character indicates the type of file, and the next nine characters indicate the permissions of the three access classes.

■ **Figure 3-1** Access classes

```
-       rwx     r-x     r--
type    user    group   other
```

The file-access permissions appear on the screen to the left of the filename when you enter the `list` command with the long option (`ls -l`):

type — The first character represents the file type. The type is not an access class, but *type* is an essential part of file permissions. In Figure 3-1, the file is a regular file (represented by –). Other file types are `d` (directory), `c` (character device), `b` (block device), `p` (FIFO or named pipe), `s` (socket), and `l` (symbolic link).

user — The next three characters in Figure 3-1, `rwx`, represent the file access permission of the owner of the file, also known as the *user.* (This owner has permission to read, write, and execute the file.) When used with `chmod` and `chgrp`, user is represented by `u`. See "Symbolic Terms," later in this chapter.

group — The next three characters in Figure 3-1, `r-x`, represent the group permissions. Any user in the same group as the file's has permission to read and execute (run) the file. Permission to write the file is left as a hyphen, meaning that the file is not changeable by the group. When used with `chmod` and `chgrp`, group is represented by `g`. See "Symbolic Terms," later in this chapter.

other — The last three characters represent the permissions for all other system users. In Figure 3-1, only `r` appears, meaning that others (who do not belong to the group and are not the owner) can read only the contents of the file. When used with `chmod` and `chgrp`, other is represented by `o`. See "Symbolic Terms," later in this chapter.

Permissions have to be set for the three access classes for each file. They can be modified manually at any time, or they can be set automatically to be the same every time a file is created.

Directory and folder permissions

Directory and folder permissions work a little differently from file permissions. Here is a list of directory and folder access permissions and their meanings:

- *Note:* A folder that you can select with a mouse is analogous to a directory that you work with at the A/UX command line. In the following discussion *directory* stands for directory or *folder.*

`r` Allows you to list filenames from the directory (`ls`).

`w` Allows you to add or delete directory entries.

`x` Allows you to search the directory, or to make it the current directory. To open a folder the permissions must be `rx`.

When you set permissions on a directory, you define who may list its contents, add or delete files in it, or change into that directory. 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 by a group member even if the files themselves deny group read permission.

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.

Additionally, 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.

Modifying a file's permissions

Only the owner of a file, or the superuser, 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 Macintosh folders. For more information, see "Directory and Folder Permissions," earlier in this chapter.

Symbolic terms

The `chmod` command can be invoked with either symbolic or numeric 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.

`chmod`

The format for invoking `chmod` with symbolic terms consists of these four arguments:

access-class — One or more of the three access classes—user (`u`), group (`g`), or other (`o`)—described in "File-Access Permissions," earlier in this chapter. In addition, the access class all (`a`) lets you grant or deny permissions to all three access classes simultaneously.

operator — Grants access permission (the + operator) or denies it (the - 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.

permissions — Read permission (`r`), write permission (`w`), and execute permission (`x`). You can grant (or deny) more than one type of permission at the same time, but you can't grant and deny permission at the same time. Also, set-uid or set-gid (`s`) and sticky bit (`t`), discussed later in this chapter in "`set-uid` and `set-gid` Commands," have symbolic terms for permissions.

filename — The file or files whose permissions are to be changed. You may use absolute or relative pathnames.

To change the permissions of a file from

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

to

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

the sequence of commands is as follows:

1. **To grant execute permission to the owner:**
   `chmod u+x` *filename*
2. **And then, to deny write and execute permissions to all others:**
   `chmod o-wx` *filename*

Numeric terms

Numeric or absolute terms are based on the combinations allowed by octal numbers where, for each access class, the mode of the file is set as follows:

0 grants no permission

1 grants execute permission

2 grants write permission

4 grants read permission

These numbers can in turn be combined in the following way:

3 (1 + 2) grants execute and write permissions

5 (1 + 4) grants execute and read permissions

6 (2 + 4) grants write and read permissions

7 (1 + 2 + 4) grants all permissions

The format for invoking `chmod` with numeric terms is
`chmod` *permission filename*

where *permission* is the numerical representation for each access class. For example, using the `chmod` command with the following numeric terms makes the file readable, writeable, and executable by the owner and group, and inaccessible to others:
`chmod 770` *filename*

The first `7` represents `rwx` for the user, the second `7` represents `rwx` for the group, and the `0` represents no access permission for all others. The permissions of the file are then

```
-rwxrwx---
```

set-uid and set-gid commands

It is possible under A/UX to set up commands that act as if they were being invoked by a specified user or by a member of a specified group. The mechanism for this is simple: a `set-uid` command takes on the user ID of its owner (the owner of the file that is being executed). The `set-gid` commands function similarly but take on the group ID of the executed file.

For example, a user might wish to change his or her password in the `/etc/passwd` file. It would normally be quite insecure to allow every user to modify the file in question, so a `set-uid` program, `passwd`, is used. When invoked, `passwd` takes on the identity of the owner of the `passwd` program, in this case `root`, for the time needed to modify `/etc/passwd`.

Of the two, `set-gid` commands tend to be safer, since group membership typically confers less power. Both should be treated with respect, however. In any case, it may be desirable to have a `set-uid` program that can be run only by a selected set of users. This can be accomplished by putting the set of users into the same group to which the program belongs and denying execute permission to others. Only group members can then run the program, performing the action as if they were the owner of the executable file.

You can use `chmod` to turn on the set-uid bit or set-gid bit for a file. You use the first field of the `chmod` command for this. The meanings of the numbers, as well as the symbolic characters (in parenthesis) that correspond to this field follow:

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:* set-uid and set-gid are applicable only with `u` or `g`.

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. In systems that load an entire file into physical memory, data is swapped in and out of memory as needed. Paging systems, however, load a page of the requested data (4K in A/UX) instead of a file at a time, which speeds data retrieval.

For additional information see Chapter 10, "System Activity Package." Note that neither set-user ID nor set-group ID modes apply to directories or nonexecutable files.

Turning on the set-uid or set-gid bit is useful with very specific and restricted files—for example, the `passwd` program. The command to turn on the set-gid bit on a file with read, write, and execute permissions for all (mode `777`) is

`chmod 2777` *filename*

The command to turn on the set-uid 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 set-gid 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 set-uid bit.

You can combine the setting of the set-uid bit and the set-gid bit, as you can with all other numeric terms, so that

`chmod 6755` *filename*

results in

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

umask and file permissions

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 you can run it individually for each user in his or her `.profile` or `.login` file. See "How A/UX Establishes the Environment," earlier in this chapter. The value assigned to `umask` in the individual files `.profile`,`.login`, or `.cshrc` overrides the values set in `/etc/profile` or `/etc/cshrc`.

The `umask` command, like the permissions associated with `chmod`, is assigned a numeric value of three octal numbers. The value of each specified digit is subtracted from the corresponding digit specified by the system for the creation of files.

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
```

so that, when the `027` is subtracted from `777`, the files' permissions are `750`. 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.

Adding a user

Adding a user to your system is a two-step process: planning the new user's working environment and then specifying it. The planning stage is important; neglecting it can lead to a very inefficient use of the system.

The manual way to add a user at the A/UX command line is described in the next section. You may want to scan this information before going to "Adding a User Quickly: `adduser`," later in this chapter. The `adduser` program automates the manual procedure. Whether you are adding one user or several at a time, you may prefer to use this program as a timesaver.

See *Setting Up Accounts and Peripherals for A/UX* for instructions on using the adduser Commando dialog box.

Adding a user manually

If you do not have an actual user now to add to your system but want to practice, use the examples given below. If you are about to add a real user, follow these steps but provide your own specifications. Before you begin, you should have a clear idea of who the user is, what his or her tasks will be, what group or groups 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.

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.

User's real full name _____

Date (year/month/day)_____

User's telephone number_____

User's login name_____

User identification number_____

Group identification number_____

Group name_____

Full pathname of the user's home directory_____

Full pathname of the user's default shell program_____

2. Pick a login name for the user.

Login names usually consist of all lowercase alphabetic characters—a maximum of 15 characters for local login or eight characters for remote logins. To make sure that the new user's login name is a new name, enter the command

`finger -m` *login-name*

This command searches the `/etc/passwd` file to see if a user already has the login name you have chosen. If you see any output, pick another login name and invoke `finger` again with the new name. If you see only the shell prompt, no one is using that login name. Enter the new login name in the form in step 1. If you are practicing, enter the name `dummy`. You could use `grep` instead of `finger`, but `finger` provides more information and also accesses the Yellow Pages database if it is in use.

3. 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 in the `/etc/passwd` file. Pick a number that is not being used and write it in the space labeled "User identification number" on the form in step 1. By convention, ID numbers under 100 are reserved for special uses, such as for special system functions.

4. **Select a group identification number.**

   If you are practicing, use 100; otherwise, see "The `/etc/group` File," earlier in this chapter, for information about selecting and specifying group membership.

5. **Select a home directory.**

   Use */dir/login-name*, where *dir* is the directory in which you are going to put the new user accounts and *login-name* is the user's login name on the form. If you are practicing, use `/users/dummy`. You may want to use a pathname such as *gm/dir/login-name*

   where *gm* represents a directory above the user's home directory. This *gm* directory should have the same group membership as the user's home directory, but the user should not have write permission on it. All users belonging to the same group should then have their home directories at the same level, that is, under *gm*. This way, the owner of the *gm* directory can be the group manager. Once you have decided who should be the group manager, write down the full pathname in the home directory space on the form. If you have a second disk, it may be useful to create a file system to hold user accounts.

6. **Select a default shell program, such as `/bin/csh` or `/bin/sh`.**

   If you have no preference, choose `/bin/csh`. You may also ask for the user's preference. See "Changing a User's Default Shell Program," later in this chapter, for information about using different command interpreters as a user's default shell program.

Specifying a user's working environment

Now that you have made your choices and have written down all the information, you can proceed with the practical steps involved in adding the user.

1. **If you are not already the superuser, log in as the root user.**

2. **Make a copy of `/etc/passwd`. For instance,**

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

   This copy is your backup in case you accidentally destroy this critical file.

3. **Next, use the `vipw` command to edit the `/etc/passwd` file. You should have all the pieces of information in front of you.**

   - *Note:* The `/etc/passwd` file is set as "read-only." The `vipw` editor copies the contents of the password file 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.

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

4. **Enter the following as the last line in the file, replacing each italicized word with the new user's information from the form you just completed.**

   login-name:*password*:*uid*:*gid*:*misc-information*:*home-directory*:*startup-program*

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

   Enter `*` in the *password* field for now. It will be filled by an encrypted version of the user's password in a few moments. The fifth field, *misc-information*, is for any miscellaneous information you care to enter (for example, the user's real name, phone number, and address). Remember to use full pathnames for the user's home directory and default shell program. If you want to play it safe, enter the following:

   ```
   dummy:*:200:100:nice guy:/users/dummy:/bin/sh
   ```

5. **Write the file and quit the editor.**

6. **Now enter the command**
   `passwd` *login-name*

   where *login-name* is the name you entered in the first field of the new entry in the `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 is too short (fewer than six characters), it asks you to enter a different password (see `passwd`(1)). Tell it only to the new user, who should log in and set a new password as soon as possible.

7. **Create the user's home directory, using the pathname you entered in field six of the new entry in the `passwd` file, with the command**
   `mkdir` *home-directory*

   If you are practicing, enter

   ```
   mkdir /users/dummy
   ```

8. **Copy the standard command files from `/usr/lib/skel`, for example:**
   `cp /usr/lib/skel/std.login` *home-dir*`/.login`

   Do the same for the `.cshrc` (C Shell), `.profile` (Bourne or Korn shell), `.kshrc` (Korn shell), and `.logout` files (C Shell). Note that the A/UX standard distribution supplies basic copies of suggested login and environment files needed for each of the A/UX shells, which are located in `/usr/lib/skel`. Use your own standard files if you have them, or edit these.

9. **Now you can 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 `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`.

   If you are practicing, change the ownership as follows:

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

10. **Next change the group membership of the user's home directory and environment by entering the commands**
    `chgrp` *group-name home-directory*
    `chgrp` *group-name home-directory/ login-files*

    where *group-name* is the name (as listed in `/etc/group`) of the group ID specified in the *GID* field of the user's entry in the `/etc/passwd` file.

    If you are practicing, enter

    ```
    chgrp project /users/dummy
    chgrp project /users/dummy/.[a-z]*
    ```

11. **Now use these commands to change the permissions associated with the user's home directory and dot files:**

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

The number `750` grants the user write, read, and execute permissions, grants members of the group read and execute permissions, and denies all permissions to other users. The number `640` grants the user write and read permissions, grants members of the group read permission, and denies all permissions to other users.

If you are practicing, change the permissions with the commands

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

For more information about these command lines, see "Modifying a File's Permissions," earlier in this chapter, and `chown`(1) and `chmod`(1) in *A/UX Command Reference.*

12. **Now log out. Let the user log in using the new login name and password and create a new file in the working environment you just established.**

If you have any problems, see "Troubleshooting," later in this chapter.

Adding a user quickly: `adduser`

A faster way to add users than that given above is to use the `adduser` program, which prompts you to enter the information necessary to create an `/etc/passwd` file entry for the user. This procedure also generates a user's password file and home directory. An entry is also made in the `/etc/group` file. Before beginning, fill out the information requested for each new user in the form shown in Step 1 of "Adding a User Manually," earlier in this chapter—unless you want the default values.

To add a user in this way, enter `adduser` at the command line and respond to these prompts:

- `user's login-name`
  Enter the user's first and last name.
- `office address/mail stop`
  Enter the information, or press RETURN to go to the next prompt.

- `office telephone`
  Enter the information, or press RETURN to go to the next prompt.
- `home telephone`
  Enter the information, or press RETURN to go to the next prompt.
- `initial group`
  The list of current groups is displayed. To create a new group, enter a new group name. The default is a new group with only this user as a member. Enter the information, or press RETURN to go to the next prompt. (A new group is assigned the next available numeric group ID.)
- `shell`
  Enter the full pathname of the shell: `/bin/csh`, `/bin/ksh`, or `/bin/sh`. The default is `/bin/csh` (C Shell).
- `home directory`
  Enter the user's full home directory path. By default, home directories are created as `/users/`*login-name*.
- `Install Useful Commands folder?`
  Enter `yes` if you want to install this folder (see *A/UX Essentials* for a description), or press RETURN to indicate no, the default.

After you have made the above entries, the system responds that the account for the *login-name* has been created. The attributes you entered are listed: login-name, user-ID, group name, if any (if the group is new, you are informed of this fact). You are then prompted:

`OK to create account?`

Enter `yes` to create it, or `no` to cancel the account.

If you create the account, you are asked:

`Require user` *login-name* `to set password on initial login?`

Enter `yes` or `no`. If you enter `no`, you are asked to set a password now.

You are then prompted to enter another account, or to press RETURN to quit the program.

The `adduser` command can also be used in batch mode, as opposed to interactively as discussed above. In this case, you enter the command and its options as described in `adduser`(1M) of *A/UX System Administrator's Reference*.

◆ *Note:* The `adduser` command does not permit new users to be added locally to a system that receives its password file through the Yellow Pages.

Modifying a user's working environment

A/UX provides great flexibility in establishing and modifying a user's working environment. Some of the more important parameters that can be modified are

- A particular user's ability to have access to the commands and data stored on the system
- The accessibility of a user's files and directories to other users
- The location or name of any user's home directory
- The command that the user employs as the shell

Distributed A/UX file permissions

The system administrator can change the permissions of system command and data files so that no users, some users, or all users can have access to them. This is a responsibility that should be exercised with extreme caution, because giving write permission to all users on a file like `/etc/passwd` can have disastrous consequences.

Users can change the permissions associated with their own files. For a discussion about how to do this and what effects these changes have on users' ability to have access to the files, see "File-Access Permissions," earlier in this chapter, and `chmod`(1) in *A/UX Command Reference.*

Moving a user

Sometimes it is necessary to move a user's working environment. There are a few ways of doing this, and the method you choose depends on the characteristics of the move. If you do move a user's files, remember to change his or her home directory in `/etc/passwd`.

Moving a directory

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-dir new-dir*

moves the *old-dir* directory (including all of its files, any subdirectories associated with it, and all of their files) to *new-dir.*

Using `cpio` to move a user across file systems

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 an 80-megabyte hard disk, the disk contains only one user-accessible file system—root. The entire A/UX directory hierarchy and any specific hierarchy (such as `/usr`) are available on this file system.

If you have created a new file system (for example, located at `/users2`) on an external hard disk, you can copy all files and subdirectories contained in the directory `/users/john` to a directory `/users2/john` on the other file system. To do so, change to the `/users` directory by entering

```
cd /users
```

and enter the following command:

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

The parts of this command line are as follows:

`find`	Name of the command that gathers the filenames to pass to `cpio`.
`john`	Name of the directory from which to start the search.
`-depth`	Forces a depth-first search of the directory in order to control the order in which files are copied.

`-print`	Prints each file or directory name found.
`|`	Connects (or "pipes") standard output of the previous command to standard input of the next command.
`cpio`	Name of the command that does the actual copying.
`-`	Character signaling that options follow.
`p`	Copies ("passes") the named files to a named directory.
`d`	Creates new subdirectories as needed.
`m`	Retains the original file's modification times.
`/users2`	Name of the new directory in which to place the files.

This moves a copy of the user's files to a new directory. Once you are sure that the move 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 files such as `.profile`.

Using `tar` to move a user across file systems

While `mv` works only within the current file system, the `tar` (tape archiver) command can 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 an 80-megabyte hard disk, the disk contains only one user-accessible file system. The entire A/UX directory hierarchy and any specific hierarchy (such as `/usr`) are available on the root file system.

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

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

The parts of this command line are as follows:

`tar`	Command name.
`c`	Creates a new `tar` image.
`f`	Stores the image under the filename (or directory name) that appears next in the line.
`-`	When used with `f`, directs the image to the standard output.
`john`	Name of the directory to start copying from.
`|`	Connects (pipes) standard output of the previous command to standard input of the next command.
`(...)`	Parentheses enclose commands to be executed in a subshell.
`cd /users2`	Changes the subshell's current directory to `/users2`.
`;`	Command separator.
`tar`	Command name.
`x`	Extracts file or files from the just-created `tar` image on tape or disk. The `tar` command does so file by file; if the file is a directory, it is extracted recursively (that is, until it is exhausted of files and subdirectories).
`f`	Extracts the image from the following file.
`-`	When used with `f`, takes the image from the standard input. When - stands for a filename, `tar` uses the standard output as a file with the `x` or `t` option.

This moves a copy of the user's files to a new directory. Once you are sure that the move was successful, 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`.

Changing a user's default shell program

The last field in the `/etc/passwd` file determines a user's default shell 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). To change a user's default shell program, all you have to do is change this field. (You or the user can also use the change shell command, `chsh`(1), to change to another one of these shells.) Other modifications to the user's working environment may be necessary, particularly with regard to shell startup files in the user's home directory; see "Files That Determine a User's Environment," earlier in this chapter.

Any program at all can serve as the default shell program. For instance, the last field of the `/etc/passwd` file can be a program such as `/bin/who`. If `who` is the default shell program, the user is able to log in but sees only the output of the program `who` before being logged out, without ever getting a shell. Although `/bin/who` is not a very useful working environment, other programs, such as the restricted shell, `rsh`(1), may be. The `rsh` program allows a user the use of a shell within the home directory but allows no directory changes.

Removing a user

Removing a user from your system may be as simple as inserting a word such as `VOID` in that user's encrypted 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 most moderate form of user removal first and then discusses additional steps that make the removal more extreme. The Macintosh interface for dragging user folders to the Trash is also discussed.

Gentle deletion

The first step in removing a user from your system is to deny the user access to it. The cleanest way to do this is to edit the user's `/etc/passwd` entry and enter the word `VOID` in the encrypted *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 is a serious security breach because anybody can log in to the system using the login name without a password.

Do not delete the whole `/etc/passwd` entry for that user yet. If you do, you will not only deny the user access to the system but also 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 names and numbers. If there is no login name for a file's owner, it is replaced by a number (when you enter `ls -l`, for instance). If you delete a few `/etc/passwd` entries, you will probably get confused about which files belong to which former user.

Backup and selective deletions

You need to be careful when deleting a user's files. 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 locate all the files owned by the user, follow these steps:

1. **Void the user's password; see the preceding section, "Gentle Deletion."**
2. **Find all the files belonging to the user, regardless of their location, with the command**
   `find / -user` *login-name* `-print >` *somefile*
3. **Back up the files using either `tar` or `cpio`, or drag them onto a floppy disk.**

   See Chapter 2, "Getting Around in A/UX," in *A/UX Essentials.* Also see the information on partial backups in Chapter 4, "Backing Up Your System."
4. **Delete the user's files after finding out if anyone is currently executing any commands or using any data files owned by that user.**

   Inquire personally or through `mail` or use the `acctcom` command (see Chapter 9, "System Accounting Package") 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 that user, remove the link. Then delete the files.

Dragging the account folder to the Trash

An alternate way to remove an account is to open the `/users` folder and drag the user's account folder to the Trash. The password file entry must then be removed as described above.

Troubleshooting

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 problem area is given first, followed by the message that the system displays, which tells you what action to take.

If the *Name* field has a name that isn't listed in `/etc/passwd`:

> Sorry, that user name is unknown. Please retype the name or contact the system administrator.

If the user's password is incorrect:

> Sorry, your password is incorrect. Please reenter it.

If the user's home directory (as listed in `/etc/passwd`) can't be found:

> 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.
>
> (Another possibility is that the system administrator made a directory in which the name differed from that in the `/etc/passwd` file.)

If the user's default shell program, for example `/bin/csh`, as listed in `/etc/passwd` can't be found:

> Your default shell program, [*name of default shell program*], does not exist. Please contact the system administrator.

If 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):

> shell program, [*name of shell program*]. Please contact the system administrator.

If the user ID (as listed in `/etc/passwd`) is out of range:

> Invalid user id [*ID number*]. Please contact the system administrator.

If the group ID (as listed in `/etc/passwd`) is out of range:

> Invalid group id [*group ID number*]. Please contact the system administrator.

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:

> Your shell program, [*name of shell program*], is not a standard shell; thus,the session type will be Console Emulator.
>
> To tell the system that this shell is a standard one, add it to `/etc/shells` or contact your system administrator.

If `/mac/bin/mac32`, or the chosen session type is missing, the following message is displayed:

> The [*kind of session*] session startup program, [*name of startup program*], does not exist. A console emulator session will be started instead.

If `/mac/bin/mac32`, or the chosen session type, is not executable by this user:

> 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.

The root user's default shell, for example`/bin/csh` as listed in `/etc/passwd`, doesn't exist:

> Your default shell program, [*name of default program*], does not exist. `/bin/sh` will be used instead.

This alert is displayed when a console message is received:

> The following console message was received: [*console message*].

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.

If the user retypes his or her password in the confirmation dialog box incorrectly, this message is displayed:

> This doesn't match your original entry. Please try again.

Only one user can change the password file at a time. Someone else may be editing it with `vipw` or the `passwd`(1) command:

> Another user is modifying the password file. Please try again later.

Chapter 4 **Backing Up Your System**

This chapter discusses the various methods by which you can **back up** your system. Backing up means that you copy the data on your hard disk to an alternate medium, such as a floppy disk or a magnetic tape, from which you can restore the data to your hard disk, if necessary. The topics covered in this chapter include:

- Full and partial backups
- Standard A/UX device files
- Mounted and unmounted file systems
- Kinds of backup media and their storage capacities
- A/UX backup utilities: `pax`, `cpio`, `tar`, `dump.bsd`, and `restore`

Making regular backup copies of files and file systems is one of the most important duties of the A/UX 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 increase your ability to restore data that is damaged, lost, or destroyed.

Store backups in a safe place—off-site if necessary. Also, keep a backup log as a written record of what 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 will be able to rebuild your system, if necessary. 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 tape cartridges, you free the system disk for use and improve performance.

The backup and `restore` utilities available on the A/UX system include `cpio`, `tar`, `pax`, and `dump.bsd`. The backup utility under the Macintosh Operating System provides an easy way to make full backups onto a 40-megabyte tape cartridge.

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 copies of files or file system data that the user stores on a removable medium, usually floppy disks or magnetic tape.

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. Although you can use `tar` and `cpio` (or `pax`) to make incremental backups, the `dump.bsd` and `restore` utilities are generally preferred. During incremental backups, the system saves its most recently modified files.

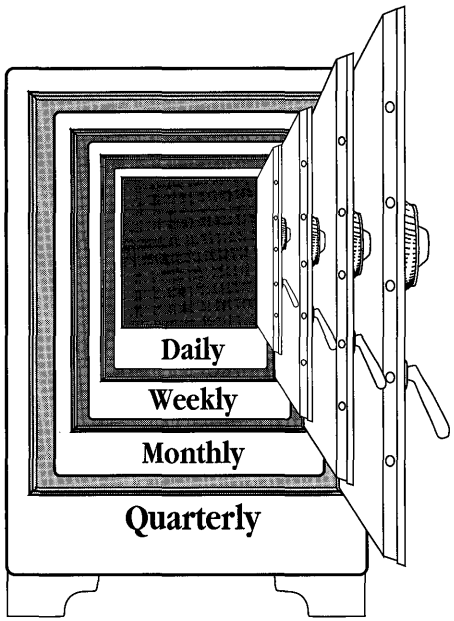
You can use the `find` command with the `-mtime` option to discover when a file was last modified, or the `-ctime` option to find out when it was last changed.

A common backup scheme

A commonly used A/UX backup scheme is illustrated in Figure 4-1, 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 4-1** 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 peripheral can be referred to by more than one device file name, as shown in Table 4-1. 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**). 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 4-1 lists the names of the device files that are standard for A/UX.

■ **Table 4-1** 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/fd/d0`	floppy disk drive #0	block
`/dev/rfd/d0`	floppy disk drive #0	char
`/dev/dsk/c8d0s0`	floppy disk drive #0	block
`/dev/rdsk/c8d0s0`	floppy disk drive #0	char
`/dev/fd/d1`	floppy disk drive #1	block
`/dev/rfd/d1`	floppy disk drive #1	char
`/dev/floppy1`	floppy disk drive #1	block
`/dev/rfloppy1`	floppy disk drive #1	char
`/dev/rdsk/c8d1s0`	floppy disk drive #1	char
`/dev/dsk/c0d0s0`	hard disk SCSI ID 0, slice 0	block
`/dev/rdsk/c0d0s0`	hard disk SCSI ID 0, slice 0	char
`/dev/rmt/tc1`	tape back up SCSI ID 1	block

◆ *Note:* A user (especially the root user) should never attempt to write data directly using the name of a device file that accesses a disk or tape drive. These filenames should be used only on backup utility command lines.

As shown in Table 4-1, you can use `/dev/dsk/c8d0s0` to refer to floppy disk drive #0. This nomenclature reflects the SCSI naming scheme, in which devices are named in the `/dev/dsk` and `/dev/rdsk` directories according to their controller, drive, and partition number. For example, `/dev/dsk/c0d0s0` signifies SCSI ID 0, the first drive, and the assigned partition. An external hard disk with SCSI ID 5 installed in the same place would be referred to as `/dev/dsk/c5d0s0`. See `gd`(7) in *A/UX System Administrator's Reference* for more information on the SCSI naming scheme. For a discussion of partition sllices, see "Making Partitions A/UX-specific: Slice Numbers" in Chapter 5.

Mounted versus unmounted file systems

A file system exists on a logical portion of a physical device. This logical portion is called a **partition** and is accessed through the device files explained in the previous section. A file system that is accessible to users is called a **mounted file system.** Except for the root file system, which is always mounted, file systems must be explicitly mounted and unmounted with the `mount` and `umount` commands.

◆ *Note:* As shipped, the only user-accessible file system on the built-in hard disk is the root file system, which is permanently mounted.

A file system is made accessible by **mounting** it on a directory, which is called the **mount point** of the file system, and can be any ordinary A/UX directory. When you change your current directory to the mount point, you may traverse the directory tree of this file system as if it were an ordinary branch of the root file system.

If you want to create a file system on a floppy disk, place a formatted disk in the drive and enter this command:

```
mkfs /dev/rfloppy0
```

To use this file system, issue the `mkdir` command to create an empty directory on the root file system, preferably with a descriptive name such as `/source`, and then use the `mount` command to mount the file system. The command sequence is

```
mkdir /source
mount /dev/floppy0 /source
```

Any files and directories you then create in `/source` appear as ordinary files and directories, even though they reside on the floppy disk drive.

To remove the file system disk, enter the commands

```
umount /dev/floppy0
eject
```

▲ **Warning** Removing the disk before issuing the `umount` command can damage the file system because A/UX may still have file system data buffered in memory. ▲

The error message

```
/source: Device busy
```

means that you or other users are accessing files or directories on the file system mounted on `/source`. If you are in single-user mode 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.

Backup media

You can make backups on a variety of media. Currently, you can back up A/UX files onto floppy disks, hard disks, cartridge tapes (using the Apple Tape Backup 40SC), or—over the network—nine-track magnetic tape. This chapter explains how to back up onto floppy disks and tape cartridges.

Storage capacity of backup media

There are two kinds of floppy disks that you can use for backing up: the 800 kilobyte single-sided disk and the 1440 kilobyte high-density disk. A 40-megabyte tape cartridge holds about 38.5 megabytes of data.

Note that use of the `tar` command does not enable storage of exactly 800K of data because of additional space that `tar` needs to keep track of directory files.

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

Refer to "Backing Up and Restoring Critical Files" in Chapter 2 of *A/UX Essentials* for instructions on preparing floppy disks for use.

When to use tape

A cartridge used by the Apple Tape Backup 40SC holds approximately 38.5 megabytes of data, compared to a high-density floppy disk, which holds 1.2 megabytes, and a regular floppy disk, which holds 800 kilobytes. Therefore, use tape cartridges when you back up large amounts of data—for example, when you make a full backup to safeguard against a system crash. Use tape cartridges when

- making weekly or monthly backups
- backing up large amounts of data, such as an entire file system

You can use `cpio`, `tar`, `pax`. and `dump.bsd`, and the Apple Tape Backup 40SC software to make backups on cartridge tape. This chapter describes how to use these A/UX utilities to back up to tape. It also discusses how to back up and restore entire disk partitions to tape.

Complete information on the Apple Tape Backup 40SC is provided in *Apple Tape Backup 40SC Owner's Guide.* Also see Chapter 6, "Adding and Managing the Apple Tape Backup 40SC," in *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.

The backup utilities

The A/UX 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 data's position 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 A/UX **copy utilities** copy data only; they operate on a single file at a time and disregard any structure that file may have (for instance, the file may be an entire file system). For information on these programs, see `dd`(1) in *A/UX Command Reference* and `volcopy`(1M) in *A/UX System Administrator's Reference.*

The A/UX archival backup utilities include `cpio`, `tar`, `pax`, and `dump.bsd`. The Apple Tape Backup 40SC software, described later in this chapter, also backs up A/UX partitions. All of the backup utilities—`cpio`, `tar`, `pax`, `dump.bsd`, and `restore`—share these advantages:

- You can 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.
- You can make full backups (of the entire file system) or incremental backups (of specific files).

pax

The `pax` utility is a new utility introduced by POSIX, a national standards organization working to find portable operating system standards that are derived from the UNIX Operating System. (POSIX stands for IEEE Standard Portable Operating System Interface for Computer Environments.) The `pax` utility reads archives created by both `tar` and `cpio`. 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 complete description.

Using cpio

The utility `cpio`, backs up and restores an entire file system or individual files. The advantages of using `cpio` include:

- It can copy device files, meaning everything stored in `/dev`, whereas `tar` cannot.
- When copying to floppy disks, `cpio` prompts you to insert another when the medium is full, and ejects the disk.

The disadvantages of using `cpio` include

- `cpio` archives are not compatible among different computer systems unless you use the `-c` option for portability.
- The `tcb` filter must be used when copying to the Apple Tape Backup 40SC drive.

The `cpio` utility copies files to or from the device or location you specify. For example, you can copy files to a disk or to a file system with equal ease.

You cannot give filenames as arguments to `cpio` as you can with `tar`. Instead, `cpio` takes its input from other utilities. The utilities most commonly used to give input to `cpio` are `ls` and `find`. Because `cpio` restores files *relative to the current directory, never* use an initial '`/`' when creating a backup with `cpio`, unless intended. Instead, move to the directory you want to copy from and then give the `cpio` command.

When using `ls` to provide input to `cpio`, do not use options to `ls` because `cpio` must receive file names at one per line.

These are most common options used with `cpio`:

`-o` Copies *out* files to a device or location you specify, such as disk, tape, 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 the Macintosh disk drive and as a result gives faster performance on a raw device than on a block device.

`cpio` and the Apple Tape Backup 40SC

When you use `cpio` to copy to tape, `cpio` cannot recognize the end of the tape. For this reason, the `cpio` command fails when it attempts to copy more files than will fit on one tape (38.5 megabytes). If the command fails for this reason, you will see the following error message:

```
tcb: No such device or address
```

Despite the wording of the error message, you need to reenter the command and specify fewer files to complete the backup successfully.

The tape unit requires that you send the data in 8 kilobyte blocks. When you use `cpio` to copy to tape, use the `tcb` filter to block the data. The sole purpose of the filter is to block data in 8 kilobyte blocks.

Here's an example:

```
ls | cpio -ov | tcb > /dev/rmt/tc1
```

The components of this command are as follows:

`ls` — The `ls` command lists the files in the current directory and sends them through a pipe as input to the `cpio` command.

`cpio -ov` — The `cpio` command, with options that copy out files (`o`), and display the filenames on the screen (`v`).

`| tcb` — Pipes the files through the `tcb` filter, which blocks the data in 8 kilobyte blocks, so that the tape unit can read it.

`> /dev/rmt/tc1` — The output device, in this case the tape controller (`tc`) for an Apple Tape Backup 40SC at (`tc`) SCSI ID 1.

Copying all files in a directory tree to a disk or tape

You may often need to copy all the files in a directory tree to a floppy disk. The simplest way to do this is with the following command:

`find` *directory-name* `-print | cpio -o > /dev/`*backupmedium*

If you are copying to tape, remember to pipe the `cpio` output through the `tcb` filter to block the data in 8-kilobyte blocks. See the preceding section, "`cpio` and the Apple Tape Backup 40SC," for further information.

Creating selective backups

To create selective backups—that is, to back up only those files that fall into a specified category—enter

`find` *directory-name* `-user` *user-name* `-print | cpio -ovB>/dev/rfloppy0`

The components of this command are as follows:

`find` *directory-name* `-user` *user-name* `-print`
: The `find` command searches the specified directory name to obtain the files of the specified user name and passes the pathnames of the user's files to `cpio`.

`| cpio -ovB`
: The `cpio` command, with options that copy out files (`o`), display the name of every file copied on the screen (`v`), and block output (`B`) of 5120 bytes per record.

`> /dev/rfloppy0`
: The output medium, in this example the (raw) disk.

You can use other `find` options to select files by other characteristics, such as group ownership or age of the file. See "Creating Incremental Backups," later in this chapter, and `find`(1) in *A/UX Command Reference.*

Creating incremental backups

To create an incremental backup, that is, to back up only files that have been modified or created since a certain time, enter

```
find / -mtime -1 -print | cpio -ovB > /dev/rfloppy0
```

The components of this command are as follows:

`find /`
: The `find` command, beginning at the root directory (`/`), passes the file pathnames to `cpio`.

`-mtime -1 -print`
: Selects files modified (`-mtime`) since the last day (`-1`) and passes the file pathnames to `cpio`. Uses `-1` for daily incremental backups and `-7` for weekly incremental backups.

`| cpio -ovB` The `cpio` command, with options that copy out files (`o`), print the name of every file copied on the screen (`v`), and block output (`B`) of 5120 bytes per record.

`> /dev/rfloppy0`
The output medium, in this example the disk.

Listing a table of contents for a disk or tape

To list the table of contents for a floppy disk or a tape made with `cpio`, enter
`cpio -it < /dev/`*backupmedium*

The components of this commands are as follows:

`cpio -it` The `cpio` command with the `-i` and `-t` flag options. The `-i` flag option specifies that input filenames should be extracted, and the `-t` option generates a table of contents.

< /dev/*backupmedium*
Uses the files on the backup medium as input to the `cpio` command.

Recovering all files on a disk or tape

To recover all files from a disk or tape created with `cpio`, enter
`cpio -ivdmu < /dev/`*backupmedium*

The components of this command are as follows:

`cpio -ivdmu` The `cpio` command with flag options. The `-i` option extracts files from the floppy disk or the tape, the `-v` option prints the filename on the screen after it has been extracted, the `-d` option creates any directories needed to extract the files, the `-m` option preserves the file's modification date, and the `-u` option extracts files from the archive unconditionally. (Normally, `cpio` does 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.

< /dev/*backupmedium*
Uses the files on the backup medium as input to the `cpio` command.

Recovering selected files from a disk or tape

To recover only certain files from a disk or tape created with `cpio`, first obtain a list of the full pathnames for files on the disk or tape, using `cpio` with the `-i` and `-t` options:
`cpio -it < /dev/`*backupmedium*

Now select the files you want to extract, and enter
`cpio -ivdmu` *filename* `< /dev/`*backupmedium*

The components of this command are as follows:

`cpio -ivdmu` The `cpio` command with flag options. The `-i` option extracts files from the floppy disk or the tape, the `-v` option displays the filename on the screen after it has been extracted, the `-d` option creates any directories needed to extract the files, the `-m` option preserves the file's modification date, and the `-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.

filename The name of the file or files to be extracted.

- *Note:* You can use file expansion characters such as `*` and `?`, but these characters must be quoted (enclosed in single or double quotes or preceded by a backslash) to prevent the shell from interpreting them before they are passed to `cpio`.

`< /dev/`*backupmedium*
Uses the files on the backup medium as input to the `cpio` 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 disks or tapes. The process must be stopped, fresh disks formatted, and the procedure started again.

Using `tar`

The utility `tar`, which stands for "tape archiver," is used to back up a directory or individual files in a file system. For complete information on this utility, refer to `tar`(1) in *A/UX Command Reference.*

One of the advantages of using `tar` is that it accepts a blocking factor (with the `b` option) that lets you match the block size of the output with the block size of the output device, such as 8-kilobyte blocks for the Apple Tape Backup 40SC.

The disadvantages of using `tar` include

- if you think your backup might not fit on one tape, disk, or other backup medium, you must specify the maximum number of blocks the medium can hold. For instance, if you do not specify blocks 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 device files.
- when copying to a floppy disk, `tar` does not eject the disk when it is full.

The `tar` command copies a single file or groups of files to or from a disk. As with `cpio`, `pax`, or `dump.bsd`, the files are copied sequentially, and no directory structure is maintained. Because `tar` restores files *relative to the current directory, never* use an initial '`/`' when creating a backup with `tar`, unless intended.

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. In general, relative pathnames are safer.

You can use `tar` to extract files copied with `tar` from the disk or tape. 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`.

The `tar` command has the capability of adding to files already on the backup medium. The `tar` command also displays a table of contents for the files archived on a particular disk or tape.

Options used with `tar` control its actions. Some options are required when copying to the Apple Tape Backup 40SC.

When copying to tape

By default, `tar` 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. However, when copying to the Apple Tape Backup 40SC, data must be sent in 8-kilobyte blocks, and this size must be specified. An example from the `/` directory follows:

```
tar cbf 16 /dev/rmt/tc1 usr/lib
```

The components of this command are as follows:

`tar cbf 16` The `tar` command with the option `cbf`. The `c` option creates a new backup, writing at the beginning of the tape. The `b` option alerts `tar` to use the next argument as the block size for sending out the files. The block size for the tape can be any multiple of 512 bytes in kilobytes. Common multiples used are 8 kilobytes and 16 kilobytes. In this example, 16 is used because blocks of 16 kilobytes are copied faster than blocks of 8 kilobyte, but this size block is not so great that a large amount of the tape is left blank if an entire block won't fit. The `f` option alerts `tar` to use the next argument, in this example `/dev/rmt/tc1`, as the place in which to copy the files.

`/dev/rmt/tc1` `usr/lib`
The device to copy to, in this case the tape unit with SCSI ID 1, and the files to copy, in this example the directory `usr/lib`.

If a backup requires multiple volumes

When copying to the Apple Tape Backup 40SC or to a floppy disk, `tar` does not recognize the end of the medium. It runs past the end of the tape or disk and generates an error message, forcing you to begin copying again. This is understandable since `tar` was designed for use with a nine-track tape, and thus its default length is 2300 feet. To protect against this annoyance, let `tar` know the holding capacity of your backup medium if you suspect that your backup might require more than one tape or disk.

Here's an example of how to let `tar` know the holding capacity of a cartridge tape:

```
tar cbBf 16 4500 /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 4500 is the maximum number of blocks the tape can hold.

The exact number of blocks a tape can hold varies, depending on the number of potentially bad blocks on the tape. It is safe to use 4500 blocks as the maximum tape capacity because this figure allows for the maximum number of bad blocks. (This figure also allows for the space required to store the tape's formatting information.) When a tape is formatted, potentially bad blocks are eliminated from the tape's usable space. If you want to know the exact capacity of a formatted tape, use the `mt` command.

Here's an example of using the `mt` command to discover the exact number of usable blocks on a tape:

```
mt -f /dev/rmt/tc1 status
```

The command returns a report that includes a line showing the number of available 8 kilobyte blocks on the tape, as shown here:

```
total  4844 blocks  (39682048 bytes)  avail this cartridge
```

Copying to a disk

You can back up A/UX files onto A/UX-formatted disks and Macintosh applications onto Macintosh-formatted disks by dragging them onto the disk. A/UX files that are copied in this way do not have permissions or dates, so this isn't recommended for commands that require this information, such as `make`.

When copying data to a floppy disk using the `tar` command, you have to enter the size of the media in 512K-size blocks (not exceeding 1600K). Since this utility prompts you to insert another disk when the medium is full but fails to ejects the disk, use `cpio` instead when you need to copy data onto more than one floppy disk.

- *Note:* Because the `tar` command writes additional data on the disk to keep track of directory files, less than the specified amount of data that a disk holds will be usable for storage.

Copying an entire directory to a disk

When copying a directory, `tar` copies all its contents, including the contents of its subdirectories.

To copy all files in the `usr/lib` directory, change to the / directory and enter the command
`tar cf /dev/`*`backupmedium`* `usr/lib`

The components of this command are as follows:

`tar cf /dev/`*`backupmedium`*
: The `tar` command with option `cf`. The `c` option creates a new backup, writing at the beginning of the disk. The `f` option uses the argument `/dev/`*`backupmedium`* to archive the files.

`usr/lib`
: The relative pathname of the directory to be copied.

- *Note:* If you plan to place these files into a location other than the one you copied them from, be sure to change to the directory / and then enter `usr/lib`. Otherwise, if you use the absolute pathname, the files will retain that pathname and must be restored to that pathname. You lose flexibility when you use absolute pathnames.

Copying specific files

You will often need to save specific files that hold important data. Such files include `/etc/passwd` and `/etc/group`. Because these files are crucial to the operation of the system, and because they are frequently modified, they are vulnerable to corruption or even loss.

To copy `/etc/passwd` and `/etc/group`, change to the / directory and enter
`tar cf /dev/`*`backupmedium`* `etc/passwd etc/group`

The components of this command are as follows:

`tar cf /dev/`*`backupmedium`*
: The `tar` command with option `cf`. The `c` option creates a new backup, writing at the beginning of the disk. The `f` option uses the argument `/dev/`*`backupmedium`* to archive the files.

`etc/passwd etc/group`
: Copies the files `etc/passwd` and `etc/group`.

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/rfloppy0 ./mvusr
```

The components of this command are as follows:

`tar rf /dev/rfloppy0`
: The `tar` command with options `rf`. The `r` option appends the file to the `tar` archive on the disk. The `f` option uses the argument `/dev/rfloppy0` as a destination for the files.

`./mvusr`
: Writes the contents of `mvusr` file from the current directory.

Adding a later version of a file to a disk or tape

To add a later version of the file `curses.mail` from the current directory to a disk or tape, enter

`tar uvf /dev/`*backupmedium* `curses.mail`

The components of this command are as follows:

`tar uvf /dev/`*backupmedium*
: The `tar` command with options `uvf`. The `u` option adds the named files to the disk or tape if they are not there or if they are modified. The `v` option displays the file size and filename. The `f` option uses the argument `/dev/`*backupmedium* to specify the name of the device on which to write the files.

`curses.mail`
: Copies the file `curses.mail` in the current directory.

The `tar` command responds with the message

```
a curses.mail 3 blocks
```

If the file has been modified, it is then copied. The `a` indicates that the file has been added to the archive. No message is printed if the file is identical to the copy in the archive.

Extracting a specific file

To recover a specific file from a floppy disk created with `tar`, use the following command. (In this example, you are recovering `chapter8`.)

`tar xf /dev/`*`backupmedium`* `chapter8`

The components of this command are as follows:

`tar xf /dev/`*`backupmedium`*
: The `tar` command with options `xf`. The `x` option extracts the specified files from the disk. The `f` option uses the argument `/dev/`*`backupmedium`* to specify the device from which to read.

`chapter8`
: The file to be extracted. Use the complete pathname.

When you use `tar` to extract a file, change directories to the target directory and specify the name as it appears in the `tar` table of contents.

Creating a table of contents from a `tar` archive

To list the files on a disk or tape created with `tar`, enter

`tar tvf /dev/`*`backupmedium`*

The components of this command are as follows:

`tar tvf`
: The `tar` command with options `tvf`. The `t` option displays a table of contents for the files on the disk. The `v` option displays the file size and filename. The `f` option uses the argument `/dev/`*`backupmedium`* to specify the device from which to read.

`/dev/`*`backupmedium`*
: The output medium holding the archive, which could be a floppy disk or tape.

The `tar` command then displays the files with their permissions, ownerships, and dates:

```
rwxr-xr-x102/202   978 Feb  1 14:16 1990    ./envelope
rwxr-xr-x102/202   211 Apr 16 11:29 1990    ./proofread
rwxr-xr-x102/202   978 May 10 10:34 1990    ./envelope
```

The same filename can appear more than once; `tar` allows multiple copies of a file on the same disk.

Recovering the latest version of a file

To recover the latest version of a file, in this example the file `curses.mail`, enter

```
tar xvf /dev/rfloppy0 curses.mail
```

The components of this command are as follows:

`tar xvf`	The `tar` command with options `xvf`. The `x` option extracts the named files. The `v` option displays verbose confirmation that the file was extracted, giving the size and name of the file. The `f` option uses the argument `/dev/rfloppy0` to archive the files.
`/dev/rfloppy0`	The output medium, in this example the disk.
`curses.mail`	The file to be extracted from the current directory.

The `tar` command then responds with the message

```
x curses.mail, 1135 bytes, 3 tape blocks
x curses.mail, 2036 bytes, 4 tape blocks
```

In this example, the `x` at the beginning of the line indicates that the file has been extracted.

Recovering a particular version of a file

To recover a particular version of a file, determine which version is needed by using the `t` option to display the contents of the disk or tape. See "Creating a Table of Contents from a `tar` Archive," earlier in this chapter.

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 /dev/`*`backupmedium`* `./mvusr`

The components of this command are as follows:

`tar xvwf /dev/`*`backupmedium`*	The `tar` command with options `xvwf`. The `x` option extracts the file from the disk, the `v` option displays the file size and filename, the `w` option causes the system to wait for user confirmation before extracting the file, and the `f` option uses `/dev/`*`backupmedium`* as the archive file.
`./mvusr`	The file to be extracted in the current directory.

When `tar` displays the correct filename, type `y`, as shown here:

```
x rwxr-xr-x 0/1    140 May  4 18:14 1990    mvusr:
x rwxr-xr-x 0/1    162 May 10 10:46 1990    mvusr:y
x ./mvusr,
162 bytes, 1 tape blocks
x rwxr-xr-x 0/1    140 May 10 10:48 1990    mvusr:
```

A `y` (yes) response indicates that the file should be extracted.

Other responses include `n` and RETURN, both indicating a "no" response.

dump.bsd and restore

The `dump.bsd` utility is used for incremental or full-file system backups. The utility supports "dump levels"; these levels range from 0 to 9, where 0 represents a full backup of the entire system and the other numbers are used for incremental backups.

The `restore` program is the companion utility of `dump.bsd`. It retrieves files and directories from a backup medium created with `dump.bsd`. When using `restore`, you must be careful not to replace the current file system with an older version of itself.

The advantages of using `dump.bsd` and `restore` include:

- These utilities are reasonably fast. Although `dump.bsd` does not make backups as fast as the Apple Tape Backup 40SC software makes an image dump, it is generally faster than `tar` and `cpio`.
- They allow you to back up only those files modified or created after a certain date. The utility does this by keeping a record of the last full dump date, and backing up only those files created or modified after that date. With `cpio` you can also do this, but it is necessary to use the `find` command to pipe the files to `cpio`.

The disadvantages of using `dump.bsd` and `restore` include:

- They operate on file systems. As distributed, A/UX has only one file system, so this is not a drawback. However, if you add one or more file systems, your use of these backup utilities becomes more complicated because you have to track 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 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, there must be an empty file with the command `/etc/dumpdates`. If this file does not exist, enter

```
touch /etc/dumpdates
```

You will not need to enter the above command again unless `/etc/dumpdates` is removed by mistake.

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.bsd` 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 0uF
  ```

- At the end of each week do a weekly dump (level 4):

  ```
  dump.bsd 4uF
  ```

- At the end of each working day do a daily dump (level 7):

  ```
  dump.bsd 7uF
  ```

The `F` on the command line tells `dump.bsd` to write the backup to the floppy disk drive (`/dev/rfloppy0`). By default, `dump.bsd` reads the files to back up from the built-in hard disk (`/dev/rdsk/c0d0s0`). Many keys can alter the default operation of the `dump.bsd` utility. See `dump.bsd`(1M) in *A/UX System Administrator's Reference.*

The level numbers mnemonically represent weeks (4 weeks in a month) and days (7 days in a week). In this strategy, the weekly dumps back up all files modified since the last monthly backup, and the daily dumps back up files modified since the last weekly dump.

For the daily and weekly dumps, you can reuse the same backup disks or tapes, overwriting your previous backups. 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).

Using `dump.bsd`

The `dump.bsd` command operates on the file system—UFS or SVFS—mounted on the specified disk partition. It copies all files modified after a certain date to a floppy disk or other backup medium.

Because many 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 end of a dump
- a hard I/O error occurs

You must answer either yes (press `y`) or no (press `n`) to any of `dump.bsd`'s questions.

The following example shows the basic format of a `dump.bsd` command:

```
dump.bsd 0uFf /dev/rfloppy0
```

This command backs up the file system on `/dev/rdsk/c0d0s0` (the built-in hard disk). The keys `0uF` have these meanings: `0` represents a complete backup (not incremental), `u` updates the system file `/etc/dumpdates` with the time of this backup, and `F` tells `dump.bsd` to write the dump on the floppy disk drive (`/dev/rfloppy0`).

dump.bsd keys

You use keys to control the operations of the `dump.bsd` command. Unless you specify at least one key, `dump.bsd` will not work. Keys are similar to flag options, except that one must be specified. These are the keys:

`F` Specifies that the dump is to be written to the disk in the floppy disk drive and sets values appropriate for dual-density disks.

- *Note:* Always specify this option, unless you are writing to an external hard disk or other external device.

`0-9` Specifies the dump level. The `dump.bsd` utility uses this number and the system file `/etc/dumpdates` to determine when the file system was last dumped (at a dump level lower than the number specified) and which files have been modified since.

```
dump.bsd 0F
dump.bsd 4F
```

In these examples, level 0 represents a full backup, whereas level 4 backs up only those files modified since a level 3 or lower-level backup.

`u` Writes the date of the beginning of the dump to the file `/etc/dumpdates`. The file records a separate date for each file system and each dump level.

```
dump.bsd 7uFf /dev/rfloppy0
```

- *Note:* The `etc/dumpdates` must exist; otherwise, `dump.bsd` will print an error message. Therefore, before you run `dump.bsd` for the first time on your system, create an empty file with the command `touch /etc/dumpdates`. You will not need to enter the above command again unless `/etc/dumpdates` is removed by mistake.

`f` *filename*

Backs up data to the specified device or file, other than the default disk.

```
dump.bsd 4uf /dev/rdsk/c5d0s0 /dev/rdsk/c0d0s0
```

In this example, the contents of `/dev/rdsk/c0d0s0` (the root file system on the internal hard disk) are written to `/dev/rdsk/c5d0s0` (in this case, an external disk).

△ **Caution**

Be very careful not to transpose the name of file being written (the first filename argument) and the name of file being read (the second filename argument). An empty file system is the likely result of such an action. △

- *Note:* If the filename is `-`, `dump.bsd` writes to the standard output, in which case it can be used as part of a pipeline.

`W`

Displays the file system that needs to be dumped. The information is gathered from the files `/etc/dumpdates` and `/etc/mtab`.

When using `W`, `dump.bsd` displays the most recent dump date and level for each file system in `/etc/dumpdates`. The file systems that need to be dumped are highlighted.

All other options are ignored when `W` is used, and `dump.bsd` exits immediately.

```
dump.bsd W
```

```
Last dump(s) done (Dump '>' file systems):
> /dev/rdsk/c0d0s0 (     /)
Last dump: Level 5,
Date Sun Nov 23 16:21
/dev/rdsk/c0d0s2 (  /usr)
Last dump: Level 0,
Date Sun Nov 30 22:01
```

In this example, the file system to be dumped is preceded by the > symbol and is not highlighted.

`w` Similar to `w`, but displays only those file systems that need to be dumped.

```
dump.bsd w
Dump these file systems:
/dev/rdsk/c0d0s0 (     /)
Last dump: Level 5,
Date Sun Nov 23 16:21
```

`n` Notifies all operators in the group `operator` that `dump.bsd` requires attention.

```
dump.bsd 0un /dev/dsk/c0d0s0
DUMP:  NEEDS ATTENTION:  Do you want to abort dump?:
("yes" or "no") yes
DUMP:  The ENTIRE dump is aborted.
```

Restoring from multiple dump levels

In the following example, the root (/) file system is restored after being removed accidentally. (The `eschatology` command in A/UX Startup enables you to boot the system.)

To restore the file system to the state in which it existed at the beginning of the month, place the first disk from the current month's level 0 dump disk in the disk drive and enter the following command:

```
restore r
```

Be aware that files and directories are recovered onto the disk relative to the current working directory.

- *Note:* If the backup archive comprises more than one disk (the usual case), `restore` will prompt you when it is necessary to insert the next disk in the series.

When the level 0 restoration is complete, place the first disk from the current week's level 4 dump disk in the drive and enter

```
restore r
```

If the level 4 backup archive comprises more than one disk, `restore` prompts you when it is ready for you to insert the next disk in the series. When the level 4 restoration is complete, the file system is restored to its state at the beginning of the week.

To complete the restoration, remove the disk from the floppy drive, place the first disk of yesterday's level 7 dump disk in the drive, and enter

```
restore r
```

When the level 7 restoration is complete, the file system is restored to its state at the time of the most recent backup (last night).

As this example shows, 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.

Using `restore`

The `restore` command reads the backup media created with the `dump.bsd` command. The following example illustrates the basic form of the `restore` command. After first changing to the mount-point directory, enter

```
restore r
```

This command reads the default disk drive (`/dev/rfloppy0`) and expects to find a disk containing a previously recorded `dump.bsd` archive. If the archive spans multiple disks (the usual case), `restore` expects to read the first disk of the archive.

The `r` key tells `restore` to load the entire contents of the archive into the current directory. (The `restore` command will recreate the entire file and directory hierarchy of the archive beginning with the current directory.)

▲ **Warning** The `r` key 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` key. If you start `restore r` from the top of a full hierarchy, you will replace current files with older versions. ▲

Like the `dump.bsd` command, the `restore` command requires keys to control its actions and accepts other arguments to specify files or directories to be restored. See "`restore` Keys," later in this chapter, for more information.

Interactive mode for `restore`

The `restore` command features an interactive mode for extracting files from a dumped disk. You can use `i` with `restore`, as in the following command:

```
restore i
```

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, selecting files to be extracted. The interface also supports commands that aid in locating files.

The commands are described here. If a command needs an argument and one is not provided, the current directory is used by default.

`ls` [*arg*] Displays the contents of the current directory or the specified directory used as its argument. In the display output

- Directory names are appended with a slash (`/`).
- Entries selected for extraction are prefixed with an asterisk (`*`).
- If the `v` key (verbose) is set, the inode number for each entry is also displayed.

`cd` *arg* Changes the current working directory to the directory specified as its argument.

`pwd`	Displays the full pathname for the current working directory.
`add` [*arg*]	Adds the current directory or specified directory to the list of files to be extracted. If a directory is used as an argument, it is recursively extracted, unless the `h` option is used in the `restore` command line.
`delete` [*arg*]	Deletes the current directory or specified directory from the list of files to be extracted. If a directory is used as an argument, it is recursively extracted, unless the `h` option is used in the `restore` command line. The easiest way to extract most files from a directory is to add the directory to the extraction list and then delete those files not needed.
`extract`	Extracts all the files on the extraction list from the backup medium. Then `restore` asks which volume you want to mount. The quickest way to extract a few files is to start with the last disk and work toward the first.
`setmodes`	In all directories added to the extraction list, the owner, modes, and times are set. If a `restore` is prematurely stopped, this option preserves the ownership, modes, and times of the directories. Nothing is extracted from the backup medium.
`verbose`	The `ls` command displays the inode number and other information for each file extracted.
`help`	Displays a list of all available commands in the interactive mode.
`quit`	Immediately terminates the `restore` program, even if all files or directories are not extracted.

`restore` keys

These are the keys most commonly used with the `restore` command. Examples illustrate how to use the keys.

`r`	Reads and loads the contents of the backup medium to the current directory. This key should be used only to restore a complete dump tape onto a clear file system, or to restore an incremental dump tape after a full level 0 dump.

- *Note:* Be very careful about where you are in the file system when you use the `r` key. If you start `restore r` from the top of a full hierarchy, you will replace current files with older versions.

`R` Used when a `restore` operation has been interrupted. It requests a particular disk from a multivolume disk set to restart a full restoration.

```
restore R
```

`x` [*arg*] Specifies files to be extracted from the backup medium.

If no file or directory is specified, `restore` begins recursively extracting from the root directory; the entire file system is extracted. If a directory name is specified, it also is recursively extracted, unless the `h` option is used.

The quickest way to extract a few files is to begin with the last disk and work toward the first. For example, if you use the command

`restore x` *filename*

only the file represented by *filename* is extracted from the backup medium. The command

`restore xh` *directory-name*

extracts files from the directory represented by *directory-name.* The command

`restore x` *directory-name*

recursively extracts the entire directory hierarchy represented by *directory-name.*

`t` [*arg*] Lists the contents of the backup medium. If a *filename* or *directory name* is used as an argument, the corresponding files that reside on the disk are listed. As with the `x` option, a directory is recursively listed, unless the `h` option is also used.

`restore` options

The `restore` command can also be used with options that accompany keys. These are the options, along with examples illustrating how to use them.

`f` *filename*

Counterpart of the `f` option to the `dump` command. The `f` option used with a *filename* argument restores data from the specified filename rather than from the default `/dev/tape`. A filename can be the name of a disk file, as shown in this example:

```
restore rf /tmp/save.level4
```

If - is used as the filename, `restore` restores from the standard input, allowing `restore` to be used as part of a pipeline.

`v`

Stands for "verbose." During a `restore` operation, the filename and file type display on the standard output.

```
restore rv
```

`y`

Causes `restore` to display a prompt asking whether to abort the restoration if a hard I/O error occurs. Otherwise, `restore` tries to skip a bad disk block and continues.

```
restore ry
```

In the event of a hard I/O error, the `restore` command responds with the message

```
Should I abort restore? yes or no
```

`h`

Extracts the actual directory and not the files that it refers to. This prevents hierarchical restoration of complete subtrees from the disk. You restore complete subtrees with the `x` key, described in the previous section, "`restore` Keys."

`restore rh` *directory-name*

Verifying data on backed-up disks

The `dd` command is used for all backup methods to find hard I/O errors. If no hard I/O error is detected, the appropriate option for each backup method can be used to display a table of contents. This action forces a read of the entire backup medium.

For example, for floppy disks, insert each disk and enter

```
dd if=/dev/rfloppy0 of=/dev/null bs=90b
```

The components of this command are as follows:

`dd`	Command name
`if=`	Input filename
`/dev/rfloppy0`	Input file (floppy disk)
`of=`	Output filename
`/dev/null`	Output file (special file used to discard output)
`bs=90b`	Sets both input and output block size to 90 blocks

If the data is successfully read, messages like the following appear:

```
17+1 blocks in
17+1 blocks out
```

If messages appear indicating hard I/O errors on any of the disks, you need to restart the entire backup, using newly formatted disks to replace the faulty ones.

The Apple Tape Backup 40SC software

An advantage of using the Apple Tape Backup 40SC software is that it is an easy way to make a full backup because it prompts you through the process with dialog boxes, and because it calculates and lets you know in advance how many tapes and minutes are required to complete the backup.

The disadvantages of using the Apple Tape Backup 40SC include

- A partition is the minimum amount of data you can retrieve.
- You need to exit A/UX to use the utility. This may be inconvenient, especially if others want to use A/UX through attached terminals.
- It makes an image backup, which copies everything on the disk, including unused parts of the disk. Therefore, it tends to require more tapes than `dump.bsd` does when backing up a disk that is less than half full.

◆ *Note:* You should generally perform backups while the system is running in single-user mode. If you make a backup of a mounted file system that is being altered by frequent writes, you risk backing up an outdated and inconsistent file system.

For instructions on using the Apple Tape Backup 40SC software, refer to Chapter 6, "Adding and Managing the Apple Tape Backup 40SC," in *Setting Up Accounts and Peripherals for A/UX.*

Chapter 5 **Preparing an Apple HD SC for A/UX**

This chapter describes how to prepare an Apple Hard Disk (HD) SC to receive an A/UX **file system.** A file system is a collection of files and file management structures on a mass storage device, such as a hard disk. (If you have a non-Apple hard disk, see the manual accompanying the disk for instructions on preparing it for use.) Preparing a hard disk simply means partitioning the disk to hold distinct types of data. Partitioning a disk is a formal way of preparing the disk to store and retrieve similar types of data and files from the same place on the disk. In nearly all cases, partitioning an Apple Hard Disk SC is done with the Apple HD 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 describes how to

- Ensure compatibility between Apple hard disk SCs, and between the disks and A/UX (see "Ensuring Apple HD SC Compatibility with A/UX")
- Recover from disk data errors by using HD SC Setup (see "Reconfiguring Partitions")
- Change the partitioning scheme of an Apple HD SC that already contains data (see "Reconfiguring Partitions")
- Use `dp` as necessary to make A/UX recognize the Misc A/UX partition created using HD SC Setup (see "Using `dp`")
- How to make and mount an A/UX file system
- Use an auxiliary hard disk partition for additional swap space (see "Adding Swap Space")

This chapter serves as a reference for the HD SC Setup commands. For complete step-by-step instructions on using HD SC Setup to partition a disk for A/UX user files, a `/usr` partition, and for A/UX files and the Macintosh OS, see *Setting Up Accounts and Peripherals for A/UX.*

Why disk subdivisions are beneficial

With Apple HD SC Setup 2.0.1, you can subdivide a hard disk into logical sections, called **partitions.** Partitions allow a disk to accommodate multiple file systems and even multiple operating systems. For example, the A/UX distribution disk contains a partition for the A/UX root file system and another for the Macintosh Operating System, which is called MacPartition. Note that a file system is not equivalent to a partition. File systems, however, are placed into partitions since this is a way of organizing the disk area so that files can be accessed easily. You can think of a partition as a part of a disk, and a file system as an organized, mountable part of a disk.

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

Multiple A/UX partitions can exist on one hard disk or on several hard disks. If you have an additional hard disk, you can place your user files in a separate partition on the auxiliary disk. It is not advisable to repartition your distribution disk.

Benefits of using HD SC Setup

The benefits of using HD SC Setup include the following:

- It allows you to create an almost unlimited variety of partitioning schemes to subdivide your disk, without having to use additional A/UX utilities, such as `dp`.
- It performs all of the arithmetic tasks necessary to arrive at the correct starting disk block for each partition, making it easier to use than the corresponding A/UX programs.
- It alters the setup information recorded on the drive to ensure compatibility between A/UX and all levels of Apple HD SC hardware currently available.

Although you can perform all partitioning and initialization functions with A/UX, only the use of HD SC Setup 2.0.1 ensures the correct operation of A/UX with all Apple HD SC hardware.

Considerations before you begin

For detailed hardware installation instructions, refer to *Apple Hard Disk SC Owner's Guide.* The most important concerns regarding the hardware setup are the following:

- Make sure the power is off while you set up the hardware.
- Set the SCSI ID number so that it doesn't conflict with existing SCSI devices.
- Position the SCSI cable terminator on the last physically connected drive in the chain.

Chapter 2, "System Startup and Shutdown," tells how to choose the startup application, which could be A/UX Startup or any other application running under the Macintosh OS.

△ **Important** The HD SC Setup program runs only under the Macintosh OS. If you need instructions on starting the system in the Macintosh OS, see *A/UX Essentials.* △

For hard disks other than Apple hard disk SCs, refer to the manufacturer's instructions. Note that HD SC Setup 2.0.1 is intended for use with Apple HD SCs only.

Ensuring Apple HD SC compatibility with A/UX

◆ *Note:* If you need HD SC Setup only to ensure Apple HD SC compatibility with A/UX, you do not need to follow any other of the procedures in this chapter.

The following steps describe how to use HD SC Setup to ensure compatibility between A/UX and all levels of Apple HD SC hardware. For example, suppose you have stored data on an Apple hard disk and you do not want to change the partitioning of the disk. In this case, you should still complete this short procedure to eliminate the possibility that A/UX is incompatible with the disk.

1. **Start or restart your computer in the Macintosh OS after inserting the Utilities 1 disk in the floppy disk drive.**

   See *A/UX Essentials* for instructions on starting the system in the Macintosh OS.

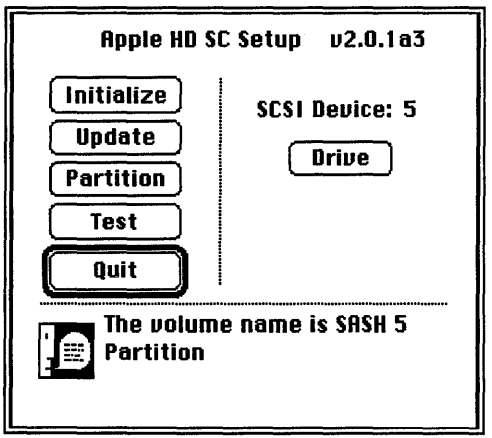
2. **Open the HD SC Setup application.**

   Double-click on the HD SC Setup icon, or click its icon to highlight it and then choose Open from the File menu. A dialog box appears, as shown in Figure 5-1.

3. **Click Quit.**

   The Macintosh desktop reappears, and the disk is now set up to be compatible with A/UX. To partition other drives, click on the Drive button, and then follow the same procedures.

■ **Figure 5-1** The main Apple HD SC Setup dialog box



Background on HD SC Setup

The HD SC Setup program is primarily used to initialize and partition disks. It was created for use with the Macintosh OS. When it is used with A/UX, many more partitioning possibilities are available.

HD SC Setup creates a partition by storing specific information in a specially reserved area on the disk, called a **partition map.** To be accessible, 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.

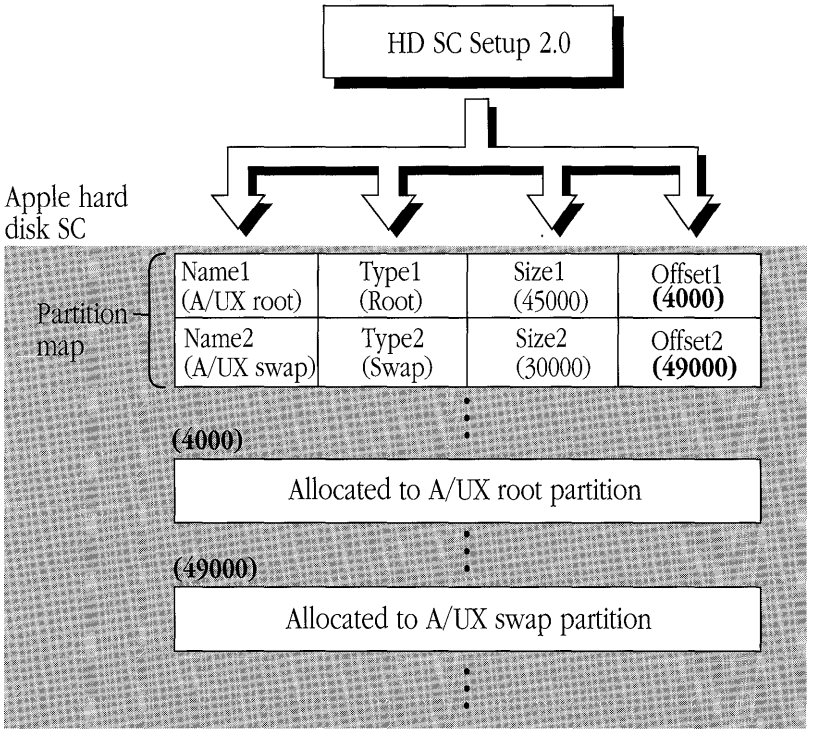
The HD SC Setup program offers a variety of functions from its opening dialog box, including disk initialization and disk partitioning. When you choose disk initialization, the system formats the disk and then tests it by writing various data patterns to all locations on the disk and verifying them. *This initialization erases all the 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, HD SC Setup builds a default partition map, indicating that one Macintosh partition of maximum size is desired.

Next, HD SC Setup asks you to provide a name for the Macintosh **volume** (the Macintosh term for a file system), within which it has automatically created and placed a file system. (This action is performed for all Macintosh partitions, which always become mountable volumes, but does not occur for any A/UX partitions.) HD SC Setup does this despite the fact that many A/UX users will not want such a partition. If you do not select the Partition button at this point and instead click Quit, this default partition map is written in the partition map on the disk. Figure 5-2 represents this process with arrows leading from HD SC Setup to the partition map fields labeled Name1, Name2, Type1, Type2, and so on. (The partition map actually includes many more fields than those shown, as described in `dpme`(4).) As shown in Figure 5-2, the offset stored in the partition map corresponds to the absolute starting disk block: Offset1 is 4000, and the A/UX root partition is likewise shown to start at that location.

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

■ **Figure 5-2** Creating disk partitions



Another way to alter the partition map is with the `dp` utility. Use it with caution since it can disrupt the partition map—for instance, by creating overlapping, unreliable partitions. Probably the only reason you would ever want to use `dp` is as a debugger, similar to `adb`, `fsdb`, and related system programmer commands. See "Using `dp`," later in this chapter, for information about using it to make small changes to a partition map.

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 an A/UX disk access in the following section.

Background on A/UX file systems

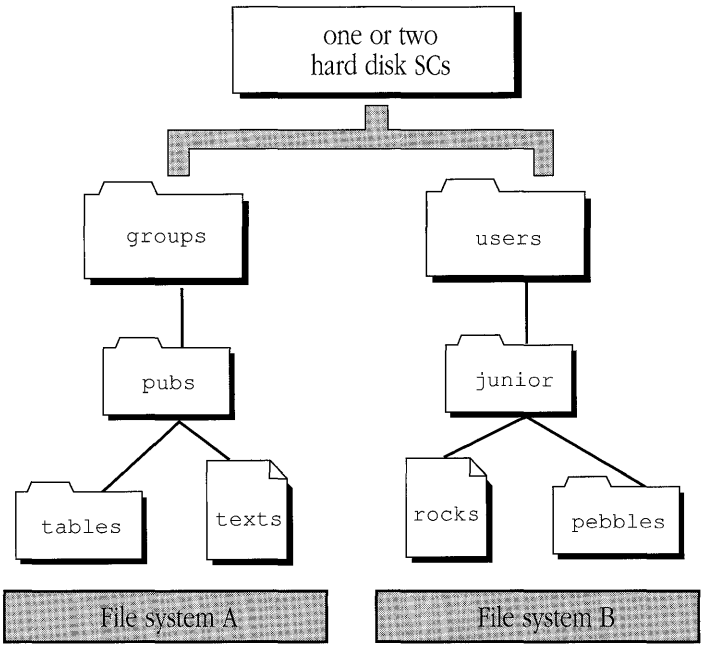
After you create partitions with HD SC Setup, the partitions exist even though there may be nothing meaningful stored in them at the outset. Nevertheless, all subsequent accesses to the disk must honor the partition boundaries recorded in the partition map. For this reason, you must handle all normal file accesses made by Macintosh or A/UX programs through the appropriate operating system routines.

A/UX interacts with partitions in a sophisticated manner, affording you many options for your partition configuration. A/UX allows multiple file systems on a single disk, and all such partitions on all disks can be simultaneously active. The Macintosh Operating System currently supports only one of its file systems per disk. For that reason, it is not worthwhile to partition a given disk with more than one Macintosh partition, although HD SC Setup allows you do so.

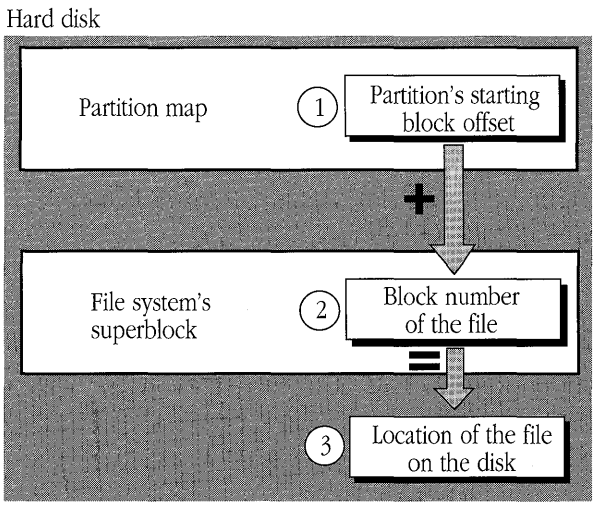
Only the administrator responsible for maintaining these configurations needs to know about the special A/UX file system administration commands. Other users can interact with the file systems transparently and do not need to know what hardware or logical partitions the operating system is accessing during a standard file access. This feature of A/UX is illustrated in Figure 5-3. As the figure shows, file systems A and B could reside together on one disk, or separately on two disks.

An A/UX **file system** is a regular data structure that can contain files and directories. It is only one kind of structure that may be placed inside of a partition. For an A/UX file system to be useful, the beginning of the file system must coincide with the beginning of a partition. Then the starting offset block of the partition can be interpreted as the first block of the file system superblock. The **superblock** is the file system header that contains information about the file system, such as size, number of inodes, and location of free blocks. Each time a file system is mounted, the kernel sets aside a block buffer to hold its superblock. This process is illustrated in Figure 5-4 as step 2 of the file-access sequence.

■ **Figure 5-3** Logical file systems compared with physical hardware



■ **Figure 5-4** File access sequence for A/UX



Following the header or superblock of a partition is a loose collection of disk blocks that can be allocated to files. A disk block is 1 kilobyte long and is usually the smallest unit of information passed to and from the hard disk. A/UX can find the disk blocks that belong to a particular file, even if they are discontinuous, because each time a block is allocated to a file, the superblock, or header, is updated to reflect these assignments. Only disk blocks that have been reserved for use by the file system are assigned to files. Thus the superblock has all the overhead information A/UX needs to alter, delete, or add files within the file system.

The three steps of a file access

Without the partition map to point out where the file system begins, A/UX cannot begin to read any of its files. See step 2 of Figure 5-4.

The first step in accessing a file is to locate the correct partition. Once the starting block offset is known, the subsequent operations are all fairly straightforward. No further disk accesses beyond the file system's own superblock are necessary to locate the file, as shown in steps 2 and 3 of Figure 5-4.

Rather than access the partition map over and over, A/UX builds a **mount table** in memory to store the currently recognized file systems. Before the mount table is updated, A/UX must follow the three-step file-access sequence shown in Figure 5-4 (the actual sequence is more elaborate). The mount table contains only the most vital information concerning partitions. For example, it does not include the partition name.

The mount table serves as a quick index to the partition map, so that the system does not have to perform step 1 of the file-access sequence.

Making partitions A/UX-specific: slice numbers

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 A/UX use (other than slice 30, which is the Macintosh volume MacPartition). A/UX locates the partitions that could contain valid A/UX file systems, or otherwise be used by A/UX, such as swap, by first reading the partition map.

A/UX **slice numbers** allow for multiple, simultaneously active, and user-configurable file systems that can be accessed through a simple user interface.

A partition that has been identified for A/UX use is assigned a slice number. The slice numbers are A/UX-specific: A/UX performs the service of translating slice numbers into the associated partition locations on the disk. If you use the Misc A/UX partition type in HD SC Setup, or if you partition the disk without HD SC Setup, then you need to use `dp` to associate a slice number with the partition.

It is not entirely correct to say that A/UX employs slice numbers to distinguish partitions. Partitions always exist. Slice numbers exist for a partition only if the partition has been recognized for A/UX use (slice numbers other than slice 30 are not associated with Macintosh partitions). Because there is no one-to-one correspondence between a slice and a partition, calling a partition a slice can create confusion. Saying that a partition has an associated slice number is more meaningful because it brings the partition into the context of A/UX.

A slice number alone is insufficient to refer to a partition. For this reason, the SCSI ID number is also used, and both are made part of a filename construct that really represents a logical disk device, as described in "Using Partition Administration Commands," later in this chapter. Only selected administrative commands require you to refer to partitions by slice numbers, since this kind of access differs from the kind that is moderated through a mount point.

When a partition contains an A/UX file system, you can use `mount`(1M) to allow normal (simplified) access to the file system. However, a slice number must be associated with the partition first.

The user's perception

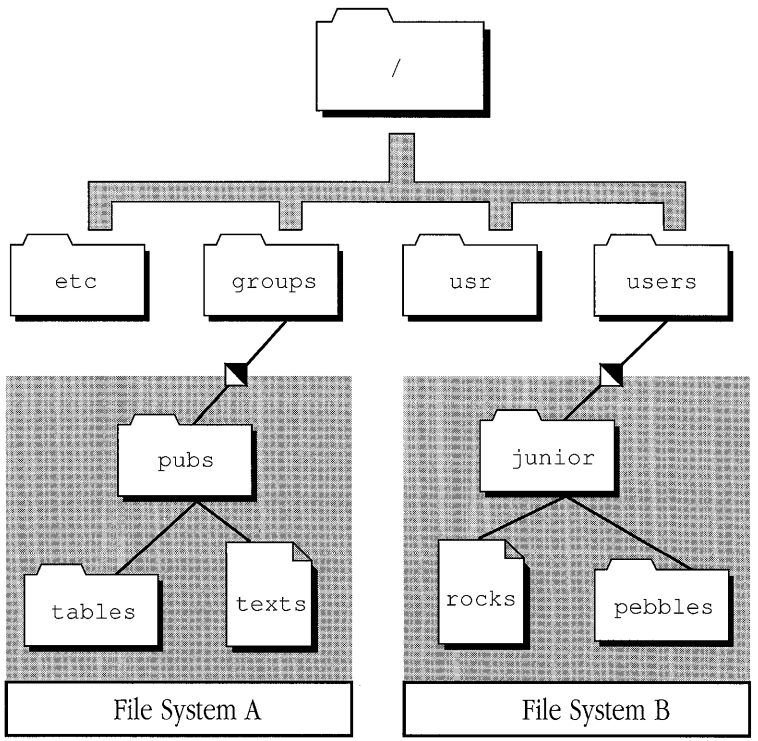
The user's view of a file system is highly regular, since most of the exceptional details about it are disguised to appear as something more familiar. This is a common occurrence in A/UX. For example, a terminal can be accessed from what appears to be a filename. A filename such as this is one of those "exceptional" objects (a terminal) that is represented as a more familiar object (a file). It not only appears as the more familiar object but also can be acted upon with many of the same commands that apply to the more familiar object.

Likewise, the directory that is the start point for another file system is treated like any other directory. A directory is analogous to a Macintosh folder, a structure that can contain other nested directories and files.

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 (the upside-down tree image). To the system administrator, the A/UX hierarchical structure is maintained by attaching file systems to other file systems at directory locations called **mount points.** Mount points are where file systems are attached to the A/UX hierarchy. A mount point is a directory that serves as a point of attachment and as a point of reference. When a mount point appears in a listing (see `ls`(1)), it seems like any other directory.

Figure 5-5 illustrates the use of mount points, in this case `/groups` and `/users`. The straight lines represent the mounting process. Before file systems are mounted on them, the `/groups` and `/users` directories would typically be empty.

■ **Figure 5-5** The mounting of file systems



Users can identify which file system a file belongs to (at least symbolically) by knowing

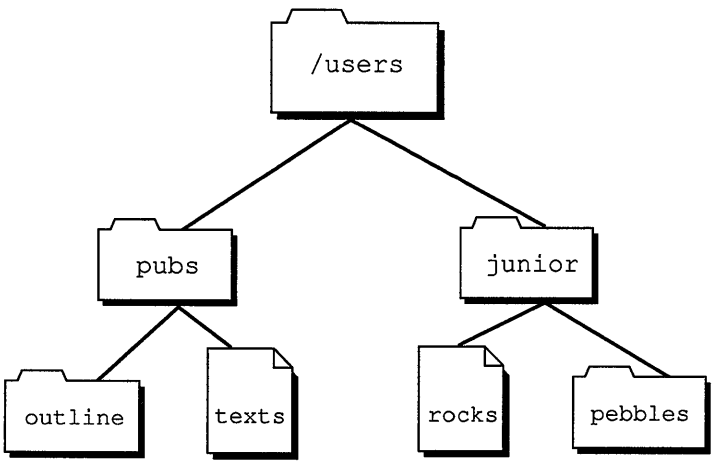
- which directories are the mount points
- where the file in question resides

The `mount`(1M) or `df`(1M) command tells you which systems are mounted and where they are mounted.

For example, the file called `texts` in Figure 5-6 is known to be in the file system mounted at `/users` because of the hierarchical relationships that exist: `texts` belongs to the directory `/pubs`, which belongs to the mount-point directory `/users`. As you would expect in a hierarchical organization, everything located under `/users` is part of the file system symbolically known as `/users`. However, you cannot tell what disk and what partition `/users` corresponds to simply from knowing that `/users` is a mount point.

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 soon becomes the name identifying the file system (see `mount`(1M)).

- **Figure 5-6** Relating a file to a mount point



The administrator's role

Most users of A/UX do not need to know where any of the mount points are, particularly if they don't have to administer the system by mounting and unmounting file systems. Because the mounting and unmounting commands are tightly secured in most UNIX implementations, special system access is usually required to perform these administrative tasks.

The mount point is what the user needs to know to gain access to the files in the file system. Even though the mount point is a symbolic name for the file system, it is the name that the user must remember. As the system administrator, you control the symbolic name for a file system simply by using the normal file renaming command, `mv`(1). (This works only while the file system is unmounted.) The symbolic way of referring to a file system is available only after you mount the file system by using the `mount`(1M) command.

The mounting process is the means by which the partition-referencing slice numbers and the file-system references to mount points become connected. The existence of a mount table entry for the mount point acknowledges that the starting block offset to a partition is known and that it should also be interpreted as the first block in a valid A/UX file system. The association of a slice number with a partition acknowledges that the starting block of a partition is known and that it is available for A/UX use. Then certain "partition-level" or "file-system-level" operations can be performed upon the associated partition (see "Using Partition Administration Commands," later in this chapter).

- *Note:* A partition that does not contain an A/UX file system (such as swap space) should never be allowed to become mounted.

The methods of 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, as opposed to a file, on a particular disk; for example, `/dev/dsk/c5d0s3`. Certain A/UX programs, such as `mount`(1M), must be able to access the partition directly.

A/UX tries to correlate partitions to slice numbers according to some simple rules. A/UX automatically gives fixed slice numbers to partitions created within HD SC Setup, as long as they are selected to be certain A/UX types of partitions (as shown in Table 5-1, later in this chapter). However, if a partition is not yet associated with a slice number, you need to use `dp` to create this association. Until a slice number identifies the partition, you cannot perform any partition administration tasks, such as creating or mounting a file system partition. Of course, if there are no slice numbers, you must assign them.

Within HD SC Setup, referring to a partition is not much of an issue for several reasons. The user need only click a screen graphic that represents all of the partitions that exist. From a system viewpoint, HD SC Setup does not actually have to make disk accesses into the partitions to create them, since the program merely updates the partition map at a known disk location.

Because HD SC Setup and A/UX use different partition referencing methods, you need to know how they relate to correlate the partitions of one to those of the other.

You may sometimes be forced to use the `dp` utility to make A/UX recognize a partition by its slice number. (Although you would need to create more than four local partitions on the root disk, or more than six on an additional disk to ever need `dp`.) The difficulty of doing this is compounded because these A/UX utilities use the partition name and the partition index number to refer to partitions. Unfortunately, HD SC Setup does not automatically show you the index number or the partition name to help you correlate what you see in HD SC Setup with what you see when you use `dp`. The partition name and the index number, also known as the partition number, are given in the Details window (see Figure 5-7). HD SC Setup does show you the partition type, which is very similar to the partition name—enough so that the partition name can be of great help when you must use `dp` to force A/UX to recognize a partition. (A procedure for doing this is described in "Using `dp`," later in this chapter.)

Using partition administration commands

By constructing the appropriate pathname, you specify which disk partition relates to the administrative operation, such as `mkfs`, `mount`, or `fsck`.

For most administrative purposes, one can refer to a particular partition on particular HD SCs according to the following pathname format:

`/dev/dsk/c`*n*`d`*m*`s`*y*

The directory `/dev` contains a list all of the devices under A/UX. The subdirectory `dsk` contains a list of the block devices that you can mount as A/UX file systems—for instance, the HD SCs.

The value of *n* is the SCSI ID of the HD SC; the value of *m* is the number of the subdrive at that SCSI ID; and the value of *y* is the slice number associated with a particular disk partition.

Some controller boards support multiple disk drives from one SCSI ID, and *m* selects that second drive. However, all of the HD SCs that you connect to the Macintosh through its built-in SCSI port have separate SCSI ID numbers; none will be identified as subdrives. For example, the internal hard disk on a computer of the Macintosh II–family or a Macintosh SE computer is identified by the device name `/dev/dsk/c0d0s`*y*; the first external HD SC—when given SCSI ID number 5, as is the convention—is identified as `/dev/dsk/c5d0s`*y*. The following is a short list of conventions that Apple currently employs:

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

The general steps in creating A/UX file systems

You may need to create A/UX file systems when you initialize a partition—that is, recreate it under HD SC Setup 2.0.1—or when you reconfigure your partitioning scheme by resizing, adding, or deleting partitions.

The complete procedure for creating A/UX file systems involves three main steps:

1. **Boot the Macintosh Operating System and partition the disk using HD SC Setup 2.0.1.**

2. **Boot A/UX and use `newfs` to create a file system for each A/UX partition in which you want an A/UX file system.**

   Use the `newfs` program described in "Using `newfs`," later in this chapter, to create a new UFS file system—the type of file system recommended for the reasons given in Chapter 1, "Managing the A/UX System: An Introduction."

3. **Mount the file system permanently with `fsentry`.**

   The `fsentry` program configures A/UX to mount the new file system each time A/UX starts up.

▲ **Warning** Be sure to use HD SC Setup 2.0.1. Earlier versions do not have all of the support capabilities discussed here. ▲

Reconfiguring partitions

The following is a list of overall steps you should complete to reconfigure partitions:

1. **Make backups of your existing file systems.**

   If you know you are going to create file systems of the same size and type when you partition the disk over again, you can use a file system utility such as `dump.bsd`(1M). (This utility creates a backup that, when it is restored, overwrites the old superblock as well as the old files.) Otherwise, use a file-oriented backup utility such as `tar`(1), `cpio`(1), or `pax`(1).

   These three utilities are more flexible than `dump.bsd`, since they allow restoration of individual files or all the files onto any valid A/UX file system with enough space.

2. **Turn to Chapter 4, "Adding and Managing Hard Disk SCs," in *Setting Up Accounts and Peripherals for A/UX* for instuctions on initializing and partitioning a disk, and creating and mounting a UFS file system.**

3. **Restore the original contents of the file system using the appropriate utility.**

Reinitializing an error-prone disk

Another situation in which you may need to create an A/UX file system is when, after trying all the less drastic methods to correct disk errors, you resort to HD SC Setup to stabilize an error-prone disk. When you normalize partitions (recreating them under HD SC Setup 2.0.1) you can use the Apple Tape Backup 40SC hardware to make tape backups of A/UX data as well as Macintosh data. The following is a list of overall steps you take to reinitialize an error-prone disk.

1. **Use the short procedure described in "Ensuring Apple HD SC Compatibility with A/UX," earlier in this chapter.**

   If you don't recover normal operations after the first try, then you don't need to repeat the procedure. If you no longer experience errors, this action alone must have been successful in restoring normal disk operation. In that case, do not perform any of the subsequent steps.

2. **If you are still experiencing disk troubles, use 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.

3. **If you are still experiencing disk troubles, you may have no choice but to 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.*

4. **Make a file system with the `newfs` command for each new partition, then run `fsentry`.**

   The `fsentry` utility makes an `fstab` entry and mounts the file system. If required, it makes a mount point. See the sections "Using `newfs`" and "Mounting a File System Permanently: `fsentry`," later in this chapter, for instructions on running this command.

5. **Restore the original contents of each of the file systems using the most recent backups available.**

Using HD SC Setup

You can use Apple HD SC Setup with any Apple SCSI hard disk—such as the Apple HD 20SC, the Apple HD 40SC, the Apple HD 80SC, or the Apple 160SC—that is connected internally or externally to a Macintosh computer. HD SC Setup provides A/UX users with an easy way to allocate space for A/UX partitions.

Setting Up Accounts and Peripherals for A/UX gives step-by-step instructions to partition a disk for

- A/UX user files only
- A `/usr` partition and A/UX user files
- A/UX user files and the Macintosh OS

This chapter provides a reference for the Details, Remove, and Group buttons. It also gives instructions for adding a new partition.

If there is no free space in which to create new partitions (there is no gray area), 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").

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.

The following steps lead you through the removal of a partition. Before you can use this subprocedure, you must have on the screen the Custom Partition dialog box.

△ **Important** Don't remove the driver unless you have a special reason. Without the driver, you won't be able to use your disk after restarting your Macintosh. △

1. **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.

2. **Click Remove.**

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

3. **Click OK.**

   Click Cancel if you decide not to remove the partition.

   When you remove a partition, the space it occupied becomes gray to represent free space. If another area of free space is adjacent, the two rectangles are combined.

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 the partitions. (For more information, see the next section, "Grouping Partitions.")

The following steps lead you through the addition of a new partition. Before you can use this subprocedure, the Custom Partition dialog box must be displayed on the screen.

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

   HD 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.

A shortcut to bypass steps 2 through 5 is to click in the top of the gray area, which presents the Partition Type dialog box. Enter the amount in the Adjust Size box and go to step 6.

2. **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.

3. **When you are satisfied with the size of the new partition, release the mouse button.**

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

   Apple HD 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.

4. **Select the partition type by clicking 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, as Table 5-1 shows.

■ **Table 5-1** A/UX partition types available

A/UX partition type	Preassigned A/UX slice number
A/UX Autorecovery	N/A
A/UX Swap	1
A/UX Root&Usr	0
A/UX Root	0
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

Note that two of the A/UX partitions are not automatically assigned a slice number. They are A/UX Autorecovery and Misc A/UX. To associate these partitions with slice numbers, you must use the A/UX `dp`(1M) utility (see "Using `dp`," later in this chapter). 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.

5. **If you wish to change the size of the partition, enter the correct size in kilobytes.**

   You can enter the partition size to a precision of a half-kilobyte (0.5K). HD SC Setup will 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 Adjust Size box; if you have selected a partition type, the minimum possible size is also shown.

6. **Click OK.**

   The HD SC Setup program creates the new partition and again presents the Custom Partition dialog box, in which the new partition is shown.

   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.

7. **Click Done in the Custom Partition dialog box to return to the main HD SC Setup dialog box.**

Grouping partitions

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 partitions combines the free space on your disk.

Sometimes this function is not available, for instance, when there are no free space areas remaining. At such times the Group button is dimmed.

Before you can use this subprocedure, you must have the Custom Partition dialog box on the screen.

1. **Click Group.**

   HD 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, HD SC Setup also warns that some information might be lost in this process.

2. **Click OK.**

   All partitions are grouped together on the disk, and they are shown together at the top of the Custom Partition display.

   Click Cancel if you decide not to group the partitions.

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.

Before you can use this subprocedure, you must have the Custom Partition dialog box on the screen.

1. **Click the partition to be moved.**
2. **Drag the partition to its new position.**

   HD SC Setup won't let you move a partition slightly into an adjacent free space. You must drag the partition more than halfway.

   When you release the mouse button, HD SC Setup presents an alert box, warning that moving information from one portion of your disk to another will take time. HD SC Setup also warns that some information might be lost in the process.

3. **Click OK to confirm.**

   Click Cancel if you decide not to move the partition.

Viewing information about partitions

Before you can use this subprocedure, you must have the Custom Partition dialog box on the screen.

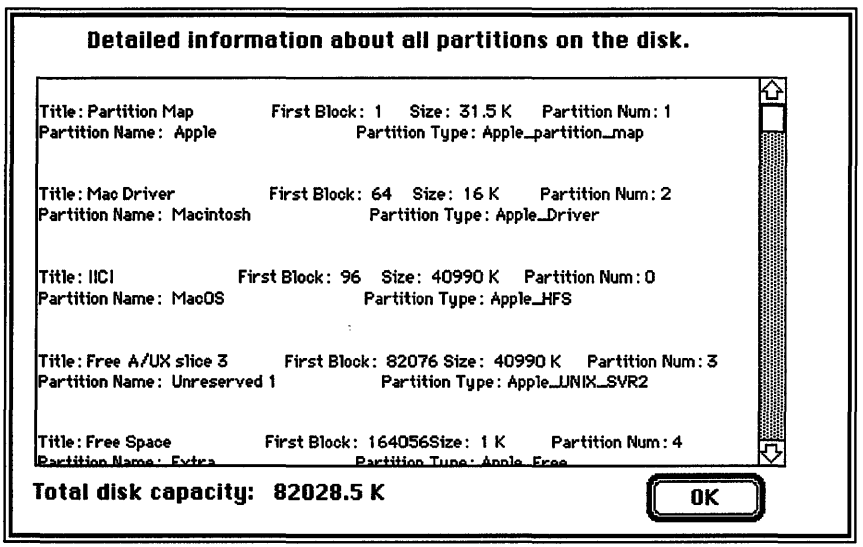
Follow these steps to see the sizes of your partitions:

1. **Click the Details button.**

   The Details window, shown in Figure 5-7, appears. It shows each partition, its name (if you must use `dp`, remember this name because you will need to use it), type, and 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, which takes up a very small portion of the disk.

■ **Figure 5-7** The Details window



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.

2. **Click OK to close the Details window and return to the Custom Partition dialog box.**

Quitting HD SC Setup

1. **Click Done in the Custom Partition dialog box to return to the main HD SC Setup dialog box.**

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

2. **Click Quit to return to the desktop.**

   No icon will show on the desktop for the new disk unless a Macintosh partition was created somewhere on the disk.

3. **Go to the section "Making and Mounting an A/UX File System," later in this chapter, or to the next section, "Using `dp`," if you chose Misc A/UX.**

Using `dp`

Perform the following procedure if you have made a partition configuration that includes any partitions of the Misc A/UX type.

- *Note:* On the boot disk, the types Root, Root&Usr, Usr, and Swap are reserved. The Swap type should be reserved for swapping on all disks. Type Autorecovery is also reserved; it doesn't have automatic slice numbers associated with it. (Table 5-1 shows how A/UX attempts to map 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.

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. You should have already created and mounted file systems with automatically mapped slice numbers. You can have up to four local partitions in the boot disk and up to six on other disks without using `dp`. (Six on other disks because types Root (slice 9) and Usr (slice 2) are reserved for the boot disk only.)

▲ **Warning** The type Autorecovery is reserved and must never be used for your own use. ▲

1. **Boot A/UX, if you have not done so already.**

   Choose Restart from the Special menu. Do not reinsert the System Tools disk. Rather, let the system boot from your installed version of A/UX.

2. **Use the `dp` utility to obtain the index numbers for any partitions with identical names, and if present, change the duplicates to unique names.**

   Replacing the drive number *n* with the SCSI ID number of the desired disk, enter the following command:

   ```
   echo P | dp -q /dev/rdsk/cnd0s31 | egrep "Index|Name"
   ```

   The `dp` utility responds with a report showing the index number and name of each partition allocated on that disk using the Apple HD 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 HD SC Setup (see Figure 5-7).

   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 HD SC, enter the following command:

   ```
   dp /dev/rdsk/cnd0s31
   ```

   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 to specify the correct SCSI ID and to type the command correctly.

Repeat the following series of substeps as many times as necessary to eliminate duplicate partition names.

- Replacing the value *x* with the index number of one of the identically named partitions, enter `c`*x*.

  You are prompted to identify the attribute field that you wish to change:
  `DPME Field?`

- Enter `n` to begin changing the partition's *name* field. Now you are prompted for a name for the partition:
  `Name [`*Old-name*`]:`

- 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?`

- 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 the substep in which you enter the index number (`c`*x*).

3. 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.

4. If you want to run file system consistency checks against the new partitions each time you reboot, 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 by entering**
   `dp /dev/rdsk/c`*n*`d0s31`

   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. **If you want information about partition 7, for example, enter**
   `p 7`

   where `p` prints the following general information about 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.

```
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` — Tells whether or not an alternate block map exists for the partition.

3. **To assign a permanent slice number for partition 7, enter the `change`, or `c`, command at the `Command?` prompt, for example:**

   ```
   c 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), the name for A/UX-specific information about the partition.**

   ```
   DPME Field? b
   ```

   You are then prompted to enter the field in the BZB that you want to modify.

5. **Enter `s` for slice number, for example:**

   ```
   BZB Field? s
   ```

6. **In response to the Slice number prompt, enter the slice number plus one (the system subtracts one from the number you enter).**

   ```
   Slice number + 1 [4]: 0
   ```

   You are then prompted to enter the next slice number to be modified.

7. **To display the current information about the BZB field, enter `p` for print at the prompt.**

   ```
   BZB Field? p
   ```

The following information is displayed:

```
No slice specified
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. **To specifiy that the BZB field be changed to slice 3, enter `s 4` (3 plus 1):**

```
BZB Field? s 4
```

9. **To display the new information about the BZB field, enter `p`.**

```
BZB Field? p
```

The following information is displayed:

```
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
No AltBlk map
```

10. **If you do not want to assign another slice number, enter `q` at the BZB and DPME field prompts:**

```
BZB Field? q
DPME Field? q
```

which returns you to the `Command?` prompt.

11. **To assign the permanent slice number, enter `w` at the `Command?` prompt to write it to disk.**

12. **To exit the program, enter `Q` at the `Command?` prompt.**

The slice number that you assigned to the disk remains until you change it again using the `dp` program.

Making and mounting an A/UX file system

If you would prefer to use the Commando dialogs instead of the command line interface to make and mount file systems, see Chapter 4, "Adding and Managing Hard Disk SCs," in *Setting Up Accounts and Peripherals for A/UX*. The number of steps given in this section for making and mounting an A/UX file system can be reduced by using the `fsentry` command or fsentry Commando dialog. See step 4.

You will probably want to store files and programs in the partitions created by HD SC Setup. To do so, you first need to create the A/UX file system structures that support them. Another possible use for partition space is as an A/UX swap area, as described in "Adding Swap Space," later in this chapter.

After partitioning your disk with HD SC Setup, you can make A/UX file systems for those partitions in which you intend to store files and programs. For each such file system, perform the following steps.

- *Note:* Some partition configurations cannot take advantage of the automatic slice number mapping. Table 5-1, earlier in this chapter, shows how A/UX attempts to map the various partition types to unique slice numbers. Until you associate these partitions with slice numbers manually, you cannot use them in the following procedure. Refer to the earlier section "Using `dp`" before continuing. You can create four user partitions on the root disk and seven on the other disks before having to use `dp`.

1. **Boot A/UX.**

   Choose Restart from the Special menu. Do not reinsert the Utilities 1 disk. Rather, let the system boot from your installed version of A/UX. The A/UX Startup application should boot A/UX to multi-user mode.

2. **Use `newfs` to make a file system inside an existing partition.**

   The `newfs` utility constructs a file system on a block device such as an HD SC. The format of the `newfs` command you will enter is

   `newfs /dev/dsk/c`*n*`d0s`*y* *type-of-disk*

The variables in this command for which you must supply values are

n The SCSI ID number of the disk. This number is normally 5 for the first external HD SC, or 6 for the second.

y The A/UX disk slice number, which identifies a particular partition.

See "Using `newfs`," later in this chapter, for more information.

3. Replace the correct values for the variables for the `newfs` command line.

If you have used the Misc A/UX type, no slice number will have been assigned. Go to "Using `dp`" earlier in this chapter to assign a slice number. Then enter the `newfs` command as described here.

`newfs /dev/dsk/c`*n*`d0s`*y* *type-of-disk*

For example, suppose that the SCSI ID number for the disk you are preparing is 6, and you chose A/UX root as the partition type. Enter the following command:

```
newfs /dev/dsk/c6d0s0 HD80SC
```

Information on the file system is displayed on the screen after the file system is created. (If you have a third-party hard disk, use the appropriate Apple disk designation; for example, `HD80SC` for an 80-megabyte hard disk.)

4. Run `fsck` on the new file system partition.

To ensure the consistency of the file system you just made, run the `fsck` utility by entering the following command, substituting the SCSI ID number of the HD SC for *n* and substituting the slice number of the partition for *y*.

`fsck -y /dev/dsk/c`*n*`d0s`*y*

For example, if the SCSI ID number of your new external HD SC 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.

5. Decide on a mount point for the new file system.

Then you can follow the rest of these steps, or use the `fsentry` command or Commando dialog to mount the file system and to make an entry in `/etc/fstab`. See "Mounting a File System Permanently: `fsentry`," later in this chapter, for details.

You attach an A/UX file system to the rest of the directory hierarchy at a mount point. 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 inacessible during that time. For example, the empty directory `/mnt` was shipped on your A/UX distribution 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 the 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 HD SC 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.

6. Mount the new file system.

Replacing the SCSI ID number of the HD SC for *n*, and replacing the partition slice for *y*, enter the following command
`mount -v /dev/dsk/c`*n*`d0s`*y* *mount-point*

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

In this case, enter

```
mount -v /dev/dsk/c6d0s0 /pubs
```

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

7. Test the file system.

To verify that your new file system is accessible, try writing to it and reading from it by giving the following commands:

```
cp /etc/passwd  /mount-point/mount.test
cat  /mount-point/mount.test
```

Your system's password file should scroll across your screen. To remove the test file, enter the command:

```
rm /mount-point/mount.test
```

If the password file does not appear, reenter the commands in this step, making sure to type them correctly. If the password file still does not appear, start over at step 1 of this section and repeat all of the steps.

8. **Update the mount entries in `/etc/fstab`.**

   You'll also want to update the file system table in `/etc/fstab`.

   When changing an important system file like this, it is always a good idea to make a copy of it first. That way, if something goes wrong, you can always reinstate the copy to its original name and restore your system to its previous state. After copying `/etc/fstab`, edit the file to include this line:

   ```
   /dev/dsk/cnd0s0 mount-point 4.2 rw 0 2
   ```

   For an explanation of the fields in `/etc/fstab`, see Figure 8-6, "A Description of Sample Entries in `/etc/fstab`." See the section "Multiple File Systems and `fsck`," in Chapter 8, for more information on the relevance of these fields. You can also refer to `fstab`(4) in *A/UX Programmer's Reference.*

Using `newfs`

The command to make a file system differs for UFS and SVFS. Use the `newfs` command to make a UFS file system. With `newfs` you don't have to enter the size of the partition in inode blocks, as required for `mkfs`; simply enter the disk type and let `newfs` calculate it for you. 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.

To run `newfs`, you must supply a slice number and SCSI ID number as part of a special filename construct. This construct identifies the partition and disk to receive the file system (see "Using Partition Administration Commands," earlier in this chapter).

The syntax for `newfs` is

`newfs /dev/dsk/c?d?s?` *device-type*

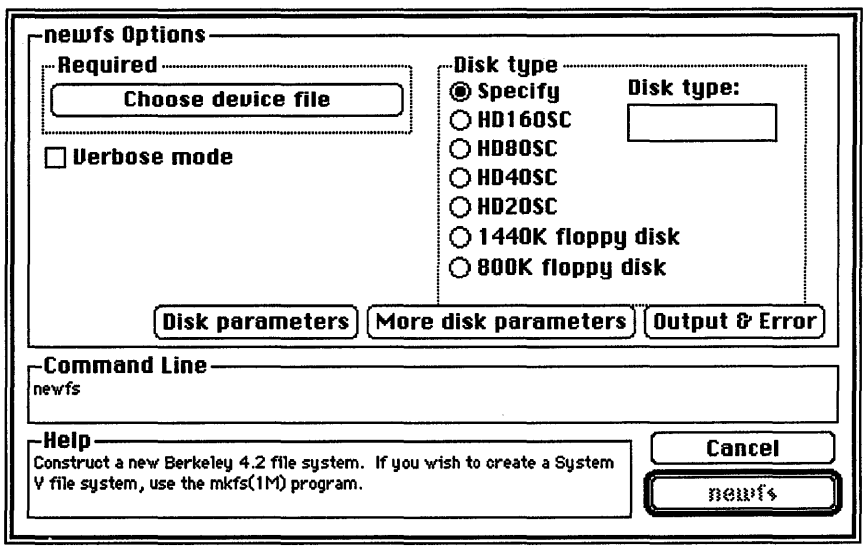
where device type can be substituted for the disk type listed in your `/etc/disktab` file—for example, `HD80SC`. If you have a third-party disk, use the Apple designation for disks of the same size.

The `mkfs` command is still supported for making SVFS file systems. To create a new SVFS file system, use the `mkfs` command, which is described in the `mkfs`(1M) man page.

◆ *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 that have become disconnected from the file system.

The `newfs` Commando dialog box shown in Figure 5-8 simplifies using this command by prompting you to build the appropriate `newfs` command line.

■ **Figure 5-8** The `newfs` Commando dialog box



Follow these steps to create a UFS file system using the newfs Commando dialog:

1. **As the root user, type `newfs` and press COMMAND-K from a CommandShell window.**
2. **Click your appropriate disk type radio button.**

   If you do not have one of those listed, click Specify and refer to your disk drive manufacturer's specifications for the correct entry.
3. **To specify the device to mount, click "Choose device file."**

   When specifying the device name for a hard disk, begin with the prefix `/dev/dsk`.
4. **Select `/dev/dsk` from the file dialog.**

   To open the `dev` folder, double-click it. To open the `dsk` folder, double-click it. A list of the extensions to the device name `/dev/dsk` is displayed. Scroll down to the one that describes your partition. Double-click it.

   The extension to a device name has the form c*n*d0s*x*, where *n* is the SCSI ID number for the disk and *x* is the slice number for the partition. If you created a partition for user files, use slice number 3. If you created a `/usr` partition, use slice number 2.

   - *Note:* If your disk constains two A/UX partitions, you'll have to run `newfs` twice. If you need to trace your way back to the `/` directory, press and drag down on the current directory name displayed at the top of the dialog box and select `/`.
5. **After selecting a name, you return to the newfs dialog box. The full device name you specified is shown in the command-line box.**
6. **Click newfs to return to the CommandShell window.**
7. **Press RETURN to run the command.**

Mounting a file system permanently: `fsentry`

Use the `fsentry`(1M) command to mount a file system and to create an entry in the file system table, `/etc/fstab`. You can make one entry each time you invoke the command. After the entry is made, the command automatically mounts the file system unless you set an option in the command line to override this action. For a description of the Commando dialog box for `fsentry`, see "Mounting a File System," in Chapter 4 of *Setting Up Accounts and Peripherals for A/UX.*

To create a file system table entry using `fsentry`, become the superuser, then enter

```
fsentry
```

at the command line, followed by these required command line arguments

`-t` *type*	The type of file system: enter `4.2` for UFS; `5.2` for SVFS (or `nfs`).
device-file	The file system to be mounted; for example, `/dev/dsk/c`*n*`d0s`*x*, where *n* is the SCSI ID number of the hard disk that contains the file system, and *x* is the slice number (usually 0 to 29, inclusive—never use slice 30 or 31).
mount-point	The full pathname of a directory on the local machine that is to be used as a mount point; for example, an A/UX user file system may be mounted at `/users` (the A/UX convention) or `/user`. The `/usr` directory tree may be mounted at directory `/usr`. The `fsentry` command creates this directory if it does not exist.

For example,

```
fsentry -4.2 /dev/dsk/c5d0s3 /user
```

creates a file system table entry that shows that the file system located at `/dev/dsk/c5d0s3` is to be mounted at `/user`.

- *Note:* If you have more than one A/UX file system on the disk, enter the `fsentry` command once for each of them.

See the `fsentry`(1M) man page in *A/UX Command Reference* for additional options that allow you to override default values. Examples are the number of passes that `fsck`, the file system checker makes; and the dump frequency used by the `dump.bsd` command discussed in Chapter 4, "Backing Up Your System."

Adding swap space

You can use the space allocated for a partition not only to create file systems but also to increase the A/UX swap area from the 18 megabytes provided. The partition that you wish to use for swap space must already be associated with Slice 1—the type Swap from HD SC Setup or the type SFS from `dp`. Refer to "Using `dp`," earlier in this chapter, to determine whether you need to perform any additional steps and what those steps are. These A/UX requirements for disk access are explained in detail in "The Three Steps of a File Access," earlier in this chapter.

Two steps are involved in using a partition as additional swap space. The first is to obtain the information you will need to specify in the `/etc/swap` command line. Once you know these details, the next step is to enter the appropriate command request containing those details.

1. **To start, obtain the starting disk block where the partition begins and the length of the partition in disk blocks.**

   Replacing the SCSI device number *n* with the correct value, enter

   ```
   echo P | dp -q /dev/dsk/cndOs31 | egrep "Name|Phys|Slice"
   ```

   Note the statistics provided for the partition you wish to be used as swap space. For example, if the partition name is Swap, the block offset to it is 108532, and its size is 19606, the following three lines of information will appear somewhere in the output of the preceding command. Note the slice number.

   ```
   Name: "Swap", Type:  "Apple_UNIX_SVR2"
   Physical: 19606 @ 108532, Logical: 19606 @ 0
   Slice 1
   ```

   - *Note:* Slice 1 is reserved for swap file systems. If you assign type Swap from HD SC Setup, or type SFS for `dp`, the slice number is Slice 1.

2. Increase your swap space.

Now you have all the important information you need for actually increasing your swap space. Replacing the SCSI device number for *n* and the slice number for *y*, enter

```
/etc/swap -a /dev/dsk/cndOsy
```

To confirm that you have done what you set out to do, obtain a report of the swap spaces currently in use by entering

```
/etc/swap -l
```

One-line descriptions of the swap areas are reported for each swap area currently recognized.

Chapter 6 **Managing Disks**

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. 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. The `autorecovery` feature, which is unique to A/UX, is discussed here. If you engage in periodic maintenance, you can use `autorecovery` to rebuild your system after a system crash.

Additionally, this chapter mentions how to access Apple's CD-ROM on a single system and over a network.

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. Before the system is booted, 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.

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 running smoothly. 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

The terms **autorecovery** and **eschatology** refer to the same A/UX feature. 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, *eschatology* is still used in filenames and command names and in the arguments to commands although it is has been replaced by autorecovery.)

One partition on the standard A/UX distribution disk is reserved for autorecovery. This area is a distinct A/UX file system 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 program 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.

At boot time, autorecovery verifies the physical condition of the disk and then checks each file in the CML. 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 add new entries to the CML and update the files in the autorecovery file system, using the autorecovery utilities eu and escher. The autorecovery program cannot function well unless a conscientious system administrator keeps the CML and autorecovery file system updated.

Using autorecovery

The autorecovery program is run from A/UX Startup. Enter the command

```
esch -b
```

For a detailed 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 was being updated when the system failed, it does not use that file system.

After checking its own file system, autorecovery checks the A/UX root file system, verifying that all blocks are readable. It marks bad blocks so that they will not be used.

The autorecovery feature then uses a version of fsck to check the A/UX root file system and each autorecovery file system for consistency. It attempts to correct any errors it finds. If a file system is not repairable, autorecovery remakes it as a last resort. When autorecovery remakes a file system, all data in the file system is lost and must be restored from backups.

The entire autorecovery process takes from 45 minutes to an hour. Most of this time is spent verifying each disk block. You can skip this phase by using the -b option of the esch command. Without this phase, autorecovery typically takes about 5 minutes, unless major system damage has occurred. See Chapter 8, "Checking the A/UX File System: fsck," for a description of the file system checking routine.

autorecovery administration

You must perform two key autorecovery administration tasks:

- Update the CML when key system files change.
- Update the autorecovery copies of key system files when they change.

You perform these tasks with the eu, escher, and eupdate utilities, described in this section and in *A/UX System Administrator's Reference.*

The utilities eu and escher perform similar functions: Both update the CML and the autorecovery file system, but they use slightly different procedures. 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 only one file at a time. To run eu, enter the command

eu *pathname*

where *pathname* is the absolute pathname of the file to be copied. An absolute pathname always begins with a slash (/).

The `eu` utility has no interactive mode. As a rule, you should use `eu` to update the CML because it creates entries with a checksum rule. The `eu` utility updates the CML even if the CML already contains an entry for the specified file.

You must be the superuser to run *eu*.

The `escher` utility

The `escher` utility adds new entries to the CML and copies the file to the `autorecovery` file system. To run `escher`, enter the command

`escher` *pathname*

where *pathname* is the absolute pathname of the file to be added.

The `escher` utility examines the specified file and adds information about the file to the CML. As discussed in the `escher`(1M) man page, `escher` does not use all possible CML rules when creating an entry; in particular, entries created with `escher` do not have a checksum rule. If you want a checksum rule, use `eu`. When the CML entry has been created, `escher` copies the file to the `autorecovery` file system.

You can also run `escher` interactively. In this mode, `escher` reads the CML and determines which files have been modified more recently than the copies in the `autorecovery` file system. For each file that `escher` finds, it prompts you to decide whether the file should be copied to the `autorecovery` file system. When the `escher` run is complete, the `autorecovery` file system is current with respect to the given files.

You must be the superuser to run `escher`.

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`, 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. You must be the superuser to run `eupdate`.

Administration guidelines

To keep your `autorecovery` file system current, follow these guidelines:

- Run `escher -m` often to obtain a list of files that may need to be updated in the `autorecovery` file system. (The list is mailed to the root account.) You can schedule regular executions of `escher -m` with a `crontab`(1) entry.
- Whenever a key file is added to the system, use the `eu` or `escher` utility to update the CML. The `autorecovery` program cannot recover files that do not appear in the CML. The file system used for `autorecovery` has severely limited disk space, so be sure that any files you add are in fact vital to `autorecovery`'s performance.
- Run `eu` or `eupdate` as soon as key system files change. For example, if you run the `autoconfig` command to build a new kernel, you should run `eupdate` as soon as you verify the operation of the new kernel.

You can use the `escher` and `eu` utilities to add entries to the CML. Currently, no utility is available to delete CML entries. Use a text editor for this task. See the `cml`(4) man page before attempting to update the CML manually.

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.

This section is organized by symptom. Error messages or warning messages may appear as part of the problem description. When appropriate, they are included here.

The following messages may appear when `autorecovery` is run from the A/UX Startup partition:

Symptom	Message
The `autorecovery` file has inconsistent mount times.	`Warning. Inconsistent mount times in bzb.`
`Autorecovery` fails during boot	`esch: no consistent type FSTEFSBZB (ES_BZBS_FSTEFS) [error occurred in Block Zero Block module]`

Occasionally, `autorecovery` may be unable to restore damaged or missing files, usually because the `autorecovery` file system was left mounted when the system was halted. The `autorecovery` program 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 the message that `autorecovery` has failed during boot. The `autorecovery` program verifies that the unmount time is later than the mount time for each `autorecovery` file system. If the `autorecovery` file system has been left mounted, this test will fail, resulting in the warning message. The `autorecovery` file system must be checked for integrity. Use the following procedure to check the offending file system.

1. **As the root user, launch `/unix` from A/UX Startup by entering**
   ```
   launch /unix
   ```
2. **At the root prompt, enter the following command (assuming that your root file system has SCSI ID 0):**
   ```
   pname -s7 "Eschatology 1"
   ```
3. **At the root prompt, enter the command**
   ```
   fsck /dev/dsk/c0d0s7
   ```
   The system checks the `autorecovery` file system for consistency.
4. **At the root prompt, enter the command**
   ```
   mount /dev/dsk/c0d0s7 /mnt
   ```

5. **Enter the command**
   `umount /mnt`

6. **Enter the command**
   `pname -u /dev/dsk/c0d0s7`

7. **Enter the command**
   `sync;sync;reboot`

   if you need to run `autorecovery` now. If you received error messages from `eu` or `escher`, you don't need to reboot.

The system will restart. The `autorecovery` program should now be operational.

The following messages may be produced when both `escher` and `eu` are preparing to update the CML at the same time:

Symptom	Message
`escher` or `eu` will not run.	`Can't lock cml file.` `eu: Can't lock the fcml. Try again later.`

The two 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 `escher` or `eu` 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 file.

The following messages may be produced when at least one device is still associated with an `autorecovery` file system:

Symptom	Message
`escher` will not run.	`/dev/dsk/cXXdYYsZZ previously pnamed` `/dev/rdsk/cXXdYYsZZ previously pnamed`

To remove the association, use the `pname` command by following these steps:

1. **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/c0d0s3: "Eschatology 1" "Apple_UNIX_SVR2"
            [not in ptab file]
   /dev/dsk/c0d0s31: "Entire Disk" "Apple_UNIX_SVR2"
            [not in ptab file]
   ```

2. **For each device associated with an `autorecovery` file system, issue the command**

   `pname -u` *device-name*

   For example, in this case, you enter

   ```
   pname -u /dev/dsk/c0d0s3
   ```

3. **Make sure that no devices are currently associated with the `autorecovery` file system by again entering**

   ```
   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.

The following message may be produced when `autorecovery` is run at boot time:

Symptom	Message
"Replaced" files missing after `autorecovery` runs.	`filename was not replaceable`

When `autorecovery` determines that a file is invalid, it 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. Instead, `autorecovery` will remove it, because it has been identified as invalid. 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 either of the `autorecovery` file systems. Alternatively, the file may exist on the `autorecovery` file system but fail the rules specified in the CML. Consistent use of the `eupdate`, `eu`, and `escher` utilities should prevent this problem.

The `autorecovery` program 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.

When run interactively, `escher` may produce the following message:

Symptom	Message
Some files cannot be copied to `autorecovery` file system.	`Ignoring filename-cmlfile entry incorrect`

The `escher` utility produces this message when the CML entry for the file to be copied contains an invalid rule specification, or when the CML fields are not separated by the correct number of tab characters. Manual editing of the CML is the most frequent cause of this condition.

To recover, first make a backup copy of the CML. Then delete the offending entry from the CML with an editor. Use `escher` or `eu` to restore the entry to the CML and copy the file to the `autorecovery` file system.

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, do 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 do not use, such as font files or games. If you're on a network, you can place read-only files, such as the online 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 the Preface for a description of each) and remove the man page files from your system. You can also condense 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. For security, be sure to make a backup copy of anything you decide to remove.

Trimming files that grow

As A/UX runs, it creates and adds to assorted log and data files. If no corrective action is taken, these files can easily grow to substantial size. Fortunately, it is easy to monitor the growth of these files, and you can use `cron` to automate much of the task of reducing their size.

Different techniques are needed for different files, however. Some files, such as `core` files, are produced by isolated system events. Once any needed information has been gained from such files, they should be removed. A one-week "cooling-off" period is usually sufficient to ensure that such a file is no longer of interest.

Other files are produced by the system to log events of administrative interest. These files should normally be trimmed of their earlier entries. The `tail` utility is particularly handy for this task.

Finally, some files must be present, but can be truncated to zero length periodically. For instance, the command `cp /dev/null foo` truncates file `foo`. Note that some log files are written into only if they exist. Removing such files causes no error condition, but disables the logging activity.

Here is a short list of system files that may need occasional action:

- `core`—memory images from damaged programs (remove)
- `/etc/wtmp`—record of spawned shells (trim)
- `/dev/*`—accidentally written data files (remove)
- `/lost+found/*`—unlinked files, salvaged by `fsck` (remove after studying)
- `/mnt/*`—misdirected files because of an unmounted file system (remove)

- /usr/adm/acct/*—output from accounting daemon (trim)
- /usr/adm/lpd-errs—output from syslog daemon (truncate)
- /usr/adm/messages—output from syslog daemon (truncate)
- /usr/adm/sulog—record of logs, moved to OLD* at reboot (trim)
- /usr/lib/cronlog—record of cron activity, moved to OLD* at reboot (trim)
- /usr/mail/*—undelivered mail (remove)
- /usr/preserve/—files from interrupted or damaged ex/vi sessions (remove)
- /usr/spool/lp/lo—output from lp (truncate)
- /usr/spool/lp/oldl—output from lp (truncate)
- /usr/spool/mqueue/syslog—output from syslog daemon (trim)
- /usr/spool/uucp/LOGFIL—record of UUCP transactions (trim)

Serving read-only files via NFS

You can use the fsentry Commando dialog to simplify the procedures for serving read-only files through the Network File System (NFS).

◆ *Note:* You must have an NFS kernel before you serve files. See *A/UX Network Administration* for instructions on setting up an NFS kernel.

The manual pages, font files, and even binary executables can be served over the Network File System, saving substantial amounts of disk space. One danger of this approach, however, is that several client machines can be adversely affected by a problem on a single server. In addition, the interactive response time of the server may be increased by the duties it performs for other machines.

In any case, before you set up a directory for shared access, you need to resolve several questions:

- Does the directory contain only shared files?
- Does the directory pose any unusual security considerations?

- Can a client system boot without access to the directory?
- Will client system responsiveness be adversely affected by network delays in accessing the files involved?
- Is the server system expected to be continuously available?
- Is use of the directory so frequent as to constitute an unreasonable drain on the server or the network itself?

Not all of these issues are absolute, and the use of multiple servers can help to share the workload, reducing the impact of a failed server. Take care, however, not to diminish the overall reliability and performance of the network.

Once the decision has been made to serve a directory, the actual procedure is simple.

1. **Add the directory to the client's `/etc/fstab` file using the fsentry Commando dialog described in Chapter 4 of *Setting Up Accounts and Peripherals for A/UX*, or by editing `/etc/fstab` with a text editor.**
2. **Add the directory to the server's `/etc/exports` file.**
3. **Try the configuration to make sure that everything is working properly.**
4. **Unmount the served directory.**
5. **Back up the contents of the directory onto disks or tapes.**
6. **Remove all files and directories under the mount point(s) of the client(s).**
7. **Remount the served directory.**

▲ **Warning** If the served directory contains binary executables, those commands will become unavailable after the client copies are removed. They will become available again only when the served copies are remounted. Therefore, make sure you have spare copies of any needed commands that will be affected by this procedure. Do not mount toolbox files or files from `/bin` or `/etc` remotely. ▲

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.

Thus only large and infrequently used data files are suitable candidates for compression. For example, in A/UX, the manual 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 poorer, 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.

Usage notes

All three 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

A word on file naming is in order at this point. 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`.

This renaming can occasionally cause unexpected and even dangerous results, however. If the input filename exceeds 12 characters, the added characters will cause the new name to exceed 14 characters. On System V file systems (SVFS), A/UX truncates any characters past the first 14, possibly overwriting the input file.

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. Each file contains an average of half a block of dead storage, because only whole blocks are allocated by the file system. The archive utilities eliminate this waste, saving considerable storage space. Note, however, that the same archiving utility will be needed after the decompression process when you unpack the archive. (The `pax` utility can be used for both; see the `pax`(1) man page.)

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` utility.

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 A.M. Also, `cron` can do things very frequently; as often as once a minute.

Once an activity has been chosen for automation via `cron`, the system administrator must decide who is to perform the action. The `cron` utility accords a command the same privileges as the user running the command. Since the superuser has so few restrictions, `cron` should typically be run from some less powerful account.

1. **The administrator must ensure that the chosen account is able to use `cron`. The file `/usr/lib/cron/cron.allow` lists the names of the users allowed to run `cron`.**

   Users can be either personal accounts, such as `joe`, or administrative logins, such as `lp`. You enter the names of the users into this file to give them access to `cron`. If this file 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`. There is little reason to use both. If neither file exists, only the superuser is allowed to use `cron`.

   In general, using the `cron.allow` file is more secure than using `cron.deny`, since the administrator must actively approve each user who might wish to use `cron`. It is more convenient, however, to use `cron.deny`, and an administrator who feels that a site's security needs are low might choose this option. Lack of both files is seldom appropriate, since all `cron` commands must then be invoked as if by the root account.

2. **Now test the desired command, using a fresh login session. Give the substitute user command (`su`) for the account that `cron` will emulate:**

   ```
   su adm
   ```

3. **Attempt to execute the desired command just as it will appear in the user's `cron` control file.**

   This ensures that the permissions, command format, and user environment are the same as those used by `cron`. (Collect multiple commands into single shell scripts, if they need to be run at the same time.)

4. **Use the `crontab`(1) command to install the new command into the `crontab` file. Use `crontab -l >`*snapshot* to get a copy of the current file.**

5. **Use `crontab` *snapshot* to edit the resulting file and then the system copy.**

   Monitor the results of a few executions of the automated activity to make sure the system is working smoothly.

CD-ROM and A/UX

A/UX supports CD-ROMs on the AppleCD SC® drive if they contain A/UX file systems. 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-ROMs on a local file system, see *Setting Up Accounts and Peripherals for A/UX.* This section describes how to use CD-ROMs to mount files over a network.

You can use a CD-ROM that already contains an A/UX file system. See *AppleCD SC Developer's Guide* for more information on creating information for a CD-ROM.

Note that A/UX does not provide support of audio CD-ROMs or CD-ROMs that use the High Sierra format or OSI9660. (The Macintosh OS does provide support for audio CD-ROMs and CD-ROM discs that use the High Sierra format.)

Follow these steps to access an AppleCD SC:

1. **Install your AppleCD SC according to the directions in *AppleCD SC Owner's Guide*.**
2. **Start up A/UX.**

   You can now access your CD-ROM just as you access a standard hard disk.

Mounting a CD-ROM as an A/UX file system

If you have a CD-ROM that contains an A/UX file system, you can mount the CD-ROM just like any other read-only file system. For example, if your AppleCD SC has SCSI ID 4, then you can insert a CD-ROM containing an A/UX file system into the drive and enter

```
mount -r /dev/dsk/c4d0s0 /cdrom
```

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 -RC /cdrom
```

You might see a listing like this:

```
./mammals:
        cats
        dogs
        elephants
        giraffes
        hippos
        lions
        mice
        squirrels
        tigers
        zebras

./programs
        prog1.c
        prog2.c
        prog3.c
        prog4.c
        prog5.c
        prog6.c
        prog7.c
        prog8.c
        prog9.c
        proga.c
```

Mounting remotely

You might want to do a remote mount of the file system on the CD-ROM, making this file system available to everyone on your network. (You must have an NFS kernel to mount a file system remotely.) For information on setting up a network, see *A/UX Network System Administration*. To allow others to do a remote mount of the A/UX file system that is on the AppleCD SC, create a mount directory for the file system, and modify `/etc/fstab` (use `fsentry` script) and `/etc/exports`. Create a mount directory for the CD-ROM, giving it a name that will be meaningful to users on other machines, for example

```
mkdir /cdrom
```

To mount the file system automatically when you enter multi-user mode, add a line for the new file system to `/etc/fstab` on the NFS server, for example

```
/dev/dsk/c0d0s0 /        ignore rw      1 1
/dev/dsk/c4d0s0 /cdrom 5.2      ro      2 2
```

The second line specifies the device with SCSI ID 4 as a type 5.2 A/UX file system that will be mounted on `/cdrom`.

Modify your `/etc/exports` file to export this file system, for example

```
/cdrom      #export to everyone on the network
```

Check that the file system is available to others by giving the command

```
showmount -e
```

You should see a line similar to

```
export list for hostname1:
/cdrom          everyone
```

To mount the file system from another machine, put an entry in `/etc/fstab` for this file system or use the `mount` command. For example, to mount the file system remotely from an NFS client machine using the mount command, enter

```
mount -o nfs,soft,ro hostname1:/cdrom /mnt
```

After remotely mounting the file system, you can access and read the files just as you would the files on any hard disk.

Chapter 7 **Managing Other Peripheral Devices**

This chapter discusses management of all peripheral devices except hard disks and CD-ROMS, which are discussed in Chapter 6, "Managing Disks." Managing peripheral devices involves connecting devices such as terminals and modems to your computer; maintaining them; and, when necessary, disconnecting them.

The first section of this chapter describes how to administer the BSD 4.3 `lpr` print spooler program, which holds print jobs when the printer is busy. The second and third sections give instructions for connecting a terminal, a modem, and any third-party devices that require kernel modification. The last section discusses the System V print spooler program—`lp`, and is provided for backward capatibility only. If you need instructions for setting up your printer, refer to Chapter 3, "Adding and Managing Printers," in *Setting Up Accounts and Peripherals for A/UX.*

All A/UX peripheral devices require a **device driver.** The ones that you connect to your machine fall into two categories: those that A/UX automatically configures with built-in device drivers and those that require adding a device driver to the A/UX kernel. A device driver is the software interface between A/UX and the peripheral device. The reason for the two categories is that the A/UX kernel has the drivers built in for commonly used Macintosh peripherals, whereas you must add device drivers for less commonly used or third-party devices. 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.

You might want to connect another computer to your A/UX system. You can do this in at least two different ways. You can connect both computers to a standard network, such as Ethernet. This is known as **networking** and is explained in *A/UX Network System Administration.* Alternatively, you can connect a computer through a serial port to the Macintosh II-family or Macintosh SE computer, and it will act as a Macintosh peripheral device. This method is discussed in "Setting Up a Terminal," later in this chapter.

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 2, because use of the `init` command may affect other users.

Using the `lpr` print spooler

This section introduces the commands necessary to use the BSD 4.3 `lpr` spooler system. A **print spooler** is a program that allows more than one person to use the same printer simultaneously, or for the same user to enter more than one job in a queue. The `lpr` system is a collection of programs and files used to manage a printer's operation. Information on `lp` is provided for backward compatibility with System V; see "System V Print Spooler: `lp`," later in this chapter.

The `lpr` system is composed of two subsystems: the spooler and the administrative system. The **spooler** intercepts print requests, schedules them for printing on a specified printer or class of printers, and then selects the appropriate interface for the printer. The **administrative system** is a series of commands you use to configure and maintain the entire `lpr` system.

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.

The `lpr` system is controlled and managed through a set of commands that

- queue or cancel requests
- query the status of requests or of the `lpr` system itself
- prevent or allow queuing requests to specific printers
- start or stop the `lpr` system
- change the printer configuration

Definitions

A definition of important terms used in this section follows:

- A **request** is a print job submitted to the `lpr` system using the `lpr` command.
- A **printer** is a unique name by which the the `lpr` 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, output is sent to the port. The `lpr` system maintains information necessary to associate each printer with a particular device. By default, the `lpr` system uses the device file `/dev/printer`, which represents the printer port—the one with the printer icon.
- The print spooler's **scheduler,** called `lpd`, 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 banners, 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

The A/UX release is shipped with the programs necessary to run the print spooler already installed and with the default AppleTalk® and ImageWriter® printer queues already created. If you have not already set up your printer, refer to Chapter 3, "Adding and Managing Printers" in *Setting Up Accounts and Peripherals for A/UX* for step-by-step instructions.

To modify your printer spooler, you must change the `printcap` file, as well as provide printer filters for any printers other than the LaserWriter® or ImageWriter. See "The `printcap` Database" and "Writing Printer Output Filters," later in this chapter, for more information.

The `printcap` database

The text file `/etc/printcap` contains entries that describe each printer controlled by the print spooler. You can edit this file using your favorite text editor. To save changes to this file, however, you must be the superuser.

This section describes some of the printer capabilities that can be defined. Refer to `printcap`(4) in *A/UX Programmer's Reference* for a definition of the format and a list of options.

The default `printcap` file includes entries for AppleTalk printers and the ImageWriter II printers connected locally via a serial line. You should only need to modify this file if you are using another type of printer.

Printer naming

Each entry in the database begins with a list of names that uniquely identify the printer described by the entry. These names may be used when invoking `lpr` from CommandShell. For example, the entry for the ImageWriter II is

```
iw|iw2|ImageWriter II:
```

To designate a particular entry as the system default destination, include the special name `lp`. The default printer in the standard configuration is the generic AppleTalk printer, which is selected using the Chooser:

```
lp|at|AppleTalk:
```

Printers on serial lines

When a printer is connected directly via a serial communication line, it must have the correct baud rate and terminal modes set.

```
:br#9600:os#0014001:cs#0004060:fd:tr=\f:
```

The `br` entry sets the baud rate for the port, and the `cs` and `os` entries set other port characteristics, which are described in `termio`(7) in *A/UX System Administrator's Reference.* The `fd` entry causes the use of hardware-supported flow control or handshaking as described in `termio`(7).

The `tr` entry indicates that a form-feed should be printed when the queue empties so that the paper can be torn off without turning the printer off-line and pressing form feed.

Since `/dev/printer` is the default port used by `lpr`, the `lp` entry is omitted here. If another *port* were to be used, it would be specified as follows:

`:lp=/dev/`*port*`:`

Spool directory

Each printer should have a separate spooling directory or else jobs will be printed on different printers, depending on which printer starts first. The following `sd` entry specifies `/usr/spool/lpd/ImageWriter` as the spooling directory (instead of the default value of `/usr/spool/lpd`):

```
:sd=/usr/spool/lpd/ImageWriter:
```

Output filters

Filters handle device dependencies and perform accounting functions. The output filter, `of`, is used when all text data must be passed through a filter. For example, the default `printcap` file includes the following (partial) description of the AppleTalk printer:

```
:of=/usr/spool/lpd/AppleTalk/ofilter:
```

Remote printers

Printers that are connected to remote hosts using the `lpr` spooler should have an empty `lp` entry. For example, this `printcap` entry sends output to the default printer on the machine *RemoteHost*:

```
remote|remote line printer:\
      :lp=:rm=RemoteHost:sd=/usr/spool/lpd/Remote
```

The `rm` entry is the name of the remote machine to connect to. This name must be a known host name for a machine on the network. If a specific printer is desired on the remote machine, the `rp` capability indicates the name of the printer.

`:rp=`*printer*`:`

Access control

The `printcap` entry `rg` controls local access to printer queues.
`:rg:` *lprgroup*

Users must be in the group *lprgroup* in order to submit jobs to the specified printer. (The default is access for all users.) Once the files are in the local queue, they can be printed locally or forwarded to another host, depending on the configuration.

Remote access is controlled by listing the hosts in either the file `/etc/hosts.equiv` or `/etc/hosts.lpd`, with one host per line. The `rsh`(1) and `rlogin`(1) commands use `/etc/hosts.equiv` to determine which hosts are equivalent for allowing logins without passwords. The file `/etc/hosts.lpd` controls which hosts have line printer access. Remote access can be further restricted to allow only remote users with accounts on the local host to print jobs through use of the `rs printcap` entry.

`lpr` commands

The commands used to administer the `lpr` system can be divided into two categories: those that any user can use, and those that only the `lpr` administrator can use. This section gives a short description of what each command does. For examples of how these commands are used, see "Printing in A/UX," in Chapter 7 of *A/UX Essentials.*

Commands for general use

`lpr` — Submits a print request to the `lpr` system. The request is printed on the default system destination or optionally routed to a specified printer or printer class. See `lpr`(1) in *A/UX Command Reference.*

`lprm` — Cancels requests by printer name or request ID number, or both, (*dest-seqno* supplied by `lpr`). Specifying the printer name cancels the job currently printing. 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 comprise a job, as well as their sizes.

Commands for `lpr` administrators

This section discusses the major commands of the `lpc` program, which provides local control over line printer activity. For information on the command format and remaining commands, refer to `lpc`(4) in *A/UX Programmer's Reference.*

`abort` and `start`

The `abort` command terminates an active spooling daemon on the local host immediately and then disables printing. It prevents new daemons from being started by `lpr`. Normally, this action forcibly restarts a frozen line printer daemon; this occurs when `lpq` reports that a daemon is present but nothing is happening. It does not remove any jobs from the queue. Use the `lprm` command to remove jobs.

The `start` command enables printing and requests `lpd` to start printing jobs.

`enable` and `disable`

The `enable` and `disable` commands turn spooling on or off in the local queue. When spooling is enabled, `lpr` can put new jobs in the spool queue; when it is disabled, `lpr` cannot add jobs. You may want to turn spooling off while testing new line printer filters, since the root user can still use `lpr` to put jobs in the queue but no one else can. 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.

`restart`

The `restart` command allows ordinary users to restart printer daemons when `lpq` reports that no daemon is present.

`stop`

The `stop` command 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.

Troubleshooting the `lpr` system

The `lpr` system error messages and possible solutions to problems are explained below. Note that the name *printer* refers to the name of the printer in the `/etc/printcap` database.

`lpr` error messages

`lpr:`*printer*`:unknown printer`

The *printer* was not found in the `/etc/printcap` file. Verify that the entry in the `/etc/printcap` file is present and correct.

`lpr:`*printer*`:jobs queued, but cannot start daemon.`

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` to make sure that it still exists. (If it doesn't, no `lpd` process will be running.) As the superuser, 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 /user/spool/lpd.lock`
```

Another possibility is that the `lpr` program is not set-user-id to root, set-group-ID to the group daemon. Check with this command:

```
ls -l /usr/ucb/lpr
```

`lpr:`*printer*`:printer queue is disabled`

This message indicates that the queue was turned off by the system administrator with command

`lpc disable` *printer*

to stop `lpr` from putting files in the queue. The system administrator can turn the printer back on by using the `lpc` command (as superuser).

lpq error messages

`waiting for` *printer* `to become ready(offline?)`

The daemon cannot open the *printer* device. 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 system device driver. Not all printers give enough information to tell you when a printer is off line or in trouble; for example, printers that are connected serially.

It is also possible that some other process, such as an output filter, has an exclusive open on the device. If this is case, use the `kill` command to end the program or programs and restart the printer with `lpc`.

printer `is ready and printing`

The `lpq` program checks to see if a daemon process exists for the *printer* and prints the file status located in the spooling directory. If the daemon is frozen, the superuser 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 is frozen and should be restarted.

`sending to` *host*

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`

The *printer* has been marked as unavailable with `lpc`.

`Warning: no daemon present`

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 terminated. The error log file for the printer and the `syslogd` logs should be checked for a diagnostic from the former process. To restart an `lpd`, enter

`lpc restart` *printer*

`lprm` error messages

`lprm:` *printer*`: cannot restart printer daemon`

This message is the same as when `lpr` prints that the daemon cannot be started.

`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 protection modes of the files are incorrect. Files may also be inaccessible if users bypass the `lpr` program when printing.

`lpc` error messages

`couldn't start printer`

This is the same as when `lpr` reports that the daemon cannot be started.

`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

The 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.

Filters are spawned by `lpd` with their standard input the data to be printed and their standard output the printer. The 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 thown away

When `lprm` sends a terminate signal to the `lpd` process controlling printing, it sends a SIGINT signal to all filters and descendents 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 its type. The `of` filter is called with the following arguments:
filter `-w`*width* `-l`*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]` `-w`*width* `-l`*length* `-i`*indent* `-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 `-x`*width* `-y`*length* `-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.

Ports

It is important that you understand the concept of a **port** when you are working with peripheral devices. A computer communicates with other equipment through a port, which is a physical connection point on your Macintosh. You can attach peripheral devices such as printers and additional terminals by connecting them (via cables or connectors) to the ports. Refer to your Macintosh *Owner's Guide* for a description of these ports and a diagram that shows their location.

There are two aspects to a peripheral connection: the hardware that connects the device to the computer and the software that allows the two to communicate. This chapter concentrates primarily on the software. For the hardware aspect of connecting a device, refer to the manual that comes with the hardware.

Look at the back of your computer and find the two serial ports. They are round and about half an inch in diameter. Each port has eight holes. The port identified by the phone icon on the back of the Macintosh computer has the A/UX device name of `/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`. You will be connecting your devices to these serial ports.

- *Note:* These ports can be switched if desired; you can attach a modem to port `tty1` and a printer to port `tty0`.

In addition to physically connecting a device to the computer, which in most cases is a rather simple operation, you must let the computer know what type of device is attached to which port and what the computer must do in each case. This is more involved, but it is not difficult.

Setting up a terminal

When you set up your computer and start it running, you'll have at least one Macintosh display working. This display runs the Finder. You can add additional terminals to interact with A/UX in command line mode as you need them. Note that only the main Macintosh display runs the Finder.

Before your Macintosh can communicate with a peripheral device, it has to know what is expected of it at each port. To do this, the computer uses the system initialization instructions found in the `/etc/inittab file 7` file and the `tty` definitions (`gettydefs`) found in the `/etc/gettydefs` file.

The `/etc/gettydefs` file contains information used by the `/etc/getty` program (the process that waits for a user to log in) to determine settings of a `getty` at a given port. The login prompt for each port is also set in this file.

The `/etc/inittab` file

The first step in telling the system about a new terminal is to change the `/etc/inittab` file. The `init` program uses this file to decide which processes to run when the system comes up. For a discussion of the `/etc/inittab` file, see Chapter 2, "System Startup and Shutdown."

- *Note:* It is a good idea to make a copy of the `/etc/inittab` file before you make any changes. This provides protection against errors when you're modifying the file.

1. **Change to the `/etc` directory and enter**
   `cp inittab inittab.old`
2. **Now open the `/etc/inittab` file using a text editor, such as TextEditor or `vi`)**

   Three of the lines in the file should look like these, though not necessarily in this order:

```
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"
```

Comments following the pound sign (#) describe the entry and, if applicable, tell how to enable it.

Each of these entries is for a given port. The entries are divided into fields, separated by colons, with the form
id:run-level:action:command

The following is a typical entry:

```
00:2:respawn:/etc/getty tty0 at_9600 # port 0
```

In this case, `/etc/getty` accepts two arguments. The first, `tty0`, specifies the port on which the command should run. The second, `at _9600`, is a label. It tells `init` to read the `/etc/gettydefs` file and use the information contained in the entry beginning with `at _9600` to set up communications with port `tty0`.

The *action* field in `/etc/inittab` entries for terminals to become active should be `respawn`. If there is anything other than `respawn` in the *action* field and you want to allow users to log in at that terminal, change this field to `respawn`. Save the `/etc/inittab` file, and enter

```
init Q
```

Setting up a serial port: `setport`

The setport Commando dialog provides a shortcut for modifying serial ports in the `/etc/inittab` file. To use the setport Commando dialog 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 at the A/UX command line.

Use the `setport` command to add or modify entries for serial ports in `/etc/inittab file 7`. The syntax is

`setport -r [-s` *speed*`]` *device-file*
`setport -o [-s` *speed*`]` *device-file*

The command's options and arguments are as follows:

device-file	The name of an existing serial port device in the `/dev` file, such as `/dev/tty0`. The `setport` command creates an entry in `/etc/inittab` for the *device-file*, if necessary. It also sets the port to allow or disallow logins.
`-r`	Sets the port to permit login sessions (respawn).
`-o`	Sets the port to disallow login sessions (off).
`-s` *speed*	Specifies the initial device speed. The default is 9600. For modems, 1200 or 2400 is usually correct.

For example,

```
setport -r -s2400 tty0
```

enables a login session on a serial port—`tty0`—with the initial speed set to 2400.

Because `setport` creates entries in `/etc/inittab`, it may be used by a device initialization routine called by `/etc/autoconfig`. Note that a device node must exist in `/dev` before running `setport`. A device node is a special file ID that A/UX uses to connect to an actual physical device. For example, the device node for serial port 0 is `/dev/tty0`. For more information about device nodes, see section 7 of *A/UX System Administrator's Reference* and the `mknod`(1M) manual page.

The `/etc/gettydefs` file

Another file that is important in managing perpipheral devices is `/etc/gettydefs`. 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 white space (blanks or tabs) between the fields for readability. The entries are of the form

label # *initial-flags* # *final-flags* # *flow-control* # *prompt* # *next-label*

where the fields are interpreted as follows:

label — String that `getty` tries to match so that it can use the entry. If the second argument to `/etc/getty` in an `/etc/inittab` entry is, for example, `co_9600`, then the entry in `/etc/gettydefs` that starts with this string is used.

initial-flags — Used to decide how the terminal is set up before login. The only critical flag at this point is the B flag, which is used to decide the communications baud (speed). In the example below, the flag is set to 9600, but it could be any valid baud rate.

final-flags — Take effect when `login` is executed. Again, speed (for instance, `B9600`) is critical. `SANE` is a composite setting; it takes care of other important terminal settings without your having to set them individually. `TAB3` specifies that tabs will be sent to the terminal as spaces. `HUPCL` specifies that the line should hang up on closing the connection. This is generally set for terminals that use a phone line through a modem. To subtract a particular attribute from a setting, prefix it with a tilde (~). Thus, `SANE ~PARENB` sets the terminal to have all the attributes of `SANE` with the exception of `PARENB` (parity). For more information on these flags, see `termio`(7) in *A/UX System Administrator's Reference* and `gettydefs`(4) in *A/UX Programmer's Reference.*

flow-control — Specifies the type of flow control to be used on the line. The settings can be `APPLE`, `DTR`, `MODEM`, and `FLOW`. Again, you can subtract a setting by prefixing it with a tilde (~).

prompt — Login prompt that appears at the terminals.

next-label — Label for `getty` to try in case the current entry causes a failure. If `getty` cannot read the keyboard input using the current definitions, it tries the `gettydef` specified in this field. For instance, if the user logs in at a terminal set up to communicate at 4800 baud but the `getty` being sent to that terminal specifies 9600 baud, the `getty` will not accept the input. When this happens, `getty` looks at this field for an alternative setting and tries entries in sequence until it finds one expecting input at 4800 baud.

Enter

```
more /etc/gettydefs
```

Two of the entries that scroll on the screen after you use this command should look something like the following, although the whole file may be several pages long.

```
co_9600 # B9600 # B9600 SANE TAB3 # ~MODEM ~DTR ~FLOW # \r\n\nApple
Computer Inc. A/UX\r\n\nlogin: # co_4800

tt_9600 # B9600 # B9600 SANE TAB3 ~MODEM ~DTR ~FLOW # \r\n\nApple
Computer Inc. A/UX\r\n\nlogin: # tt_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.

Using another computer as a terminal

See *Setting Up Accounts and Peripherals for A/UX* for the setport Commando dialog that simplifies this procedure. (The `setport` command is discussed earlier in this chapter in "Setting Up a Serial Port: `setport`.") This section discusses how to set up a terminal by using the command line interface to edit the `/etc/inittab` file.

It is possible to connect another computer to a system just as a terminal is attached, through a serial line. For example, you can treat a Macintosh computer running the MacTerminal® application exactly like a terminal. As long as the files `/etc/inittab` and `/etc/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 Plus or Macintosh SE 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 mini-8 connectors at both ends.

△ **Important** Be sure that the power for both machines 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. **Log in to A/UX on your system as the root user.**

   If the root command prompt appears on the console, you're already logged in as the root user.

4. **Make a copy of `/etc/inittab`.**

   When you receive the root command prompt after finishing step 3, enter

   ```
   cp /etc/inittab /etc/inittab.old
   ```

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.

- *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.

5. **Use your favorite text editor to edit `/etc/inittab`.**

   Change the line for `tty0` to replace `off` with `respawn` so that the beginning of the line looks like this:

   ```
   00:2:respawn:/etc/getty tty0 at_9600
   ```

6. **At the command line prompt enter `init q` to effect the changes in `/etc/inittab`.**

   ```
   init q
   ```

7. **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 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 serve the terminal.

   If you receive no output from the command, or if a number other than `0` appears in the third column, enter the following command:

   ```
   cp /etc/inittab.old /etc/inittab
   ```

   After entering this command, begin these instructions over again at step 5.

8. **Log in to A/UX from the Macintosh Plus.**

   You should now have a login prompt on the Macintosh Plus. Log in as the root user or any other valid user to test the connection. Enter your password when prompted, and press RETURN to accept the VT100™ terminal type. Your Macintosh Plus is now serving as a functioning terminal for A/UX.

If you don't get a login prompt on the Macintosh Plus, check the cable connection or your settings on the terminal-emulation application. If the command in step 7 was successful, the problem is not likely to be related to A/UX.

9. **Configure your terminal-emulator application on the Macintosh Plus.**

   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 (The VT100, a popular terminal manufactured by Digital Equipment Corporation, is 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

Attaching a VT100, VT100 emulator, or other terminal

You can attach a VT100 or a VT100 emulator or other terminal to your Macintosh, which enables you to interact with A/UX at the command line. After physically connecting the terminal, run the `setport` command, which enables the terminal entry in the `/etc/inittab` file. These steps are sufficient unless your terminal is *not* a VT100 or VT100 emulator.

To attach non-VT100 terminals, perform the following steps:

1. **As the superuser, edit the `/etc/ttytype` file using TextEditor or `vi`.**
2. **Replace VT100 in the following line:**
   `VT100  tty1`

   (where `tty1` is the same entry as you enabled in the `/etc/inittab` file)

   with the designation of the terminal, for example WYSE350.

   To determine the name of the terminal, display the `/etc/termcap` file. If your terminal does not exist as an entry, consult your terminal's manual or vendor for an entry in the file that matches yours or that you can modify. Use this designation for `tty1`. (Information about setting your terminal's parity bits and baud rates can be supplied by your vendor.)

Setting up a modem

See *Setting Up Accounts and Peripherals for A/UX* for a discussion of the setport Commando dialog that simplifies this procedure. (The `setport` command is discussed earlier in this chapter in "Setting Up Serial Ports: `setport`.") This section describes how to set up a modem using the A/UX command line interface.

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` on the line. You can set up a dial-out modem on one port and a dial-in modem on the other port. This section uses the examples of a modem on port `tty0` and a terminal on port `tty1`. This is arbitrary. You can put the modem on either port.

◆ *Note:* You cannot have incoming and outgoing calls simultaneously on the same modem.

Setting up an Apple Personal Modem

Setting up an Apple Personal Modem requires setting the baud rate and running a program that sets the modem for auto-answer dial-in. Both modifications are accomplished with changes to `/etc/inittab`. To set the Apple Personal Modem for auto-answer and dial-in, and set the baud rate, change the `/etc/inittab` entry from

```
00:2:off:/etc/getty tty0 at_9600
```

to

```
00:2:respawn/etc/apm_getty tty0 mo_1200
```

For dial-in lines, `/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`.

Dial-out access only

Suppose the `/etc/inittab` file has the following entries:

```
00:2:respawn:/etc/getty tty0 at_1200 # Port tty0
01:2:respawn:/etc/getty tty1 at_9600 # Port tty1
```

and you want the modem on port `tty0` to work as an outgoing device only.

1. **Find the line**

   ```
   00:2:respawn:/etc/getty tty0 at_1200 # Port tty0
   ```

   and change the line to

   ```
   00:2:off:/etc/getty tty0 at_1200 # Port tty0
   ```

2. **Now that you've made the necessary preparations, you're ready to use your modem.**

   To allow outgoing calls, enter `init q`, or kill the `getty`. When the system is in multi-user mode, the new entry in `/etc/inittab` takes effect, and your modem functions in outgoing mode.

3. **To make a call, you need to know the phone number of a modem attached to another computer set up to receive calls.**

   Once you have a number, you can use the `cu` command (among others) to get your modem to talk to other computers. This is the command used in this manual for testing purposes.

   Most modems operate at 300 or 1200 baud (or higher). Using a modem at 300 baud can be extremely slow. If possible, operate the modem at 1200 baud. You can do this using the `cu` flag option `-s`:

   `cu -s`*speed*

   In this case,

   ```
   cu -s1200
   ```

   Another key to getting `cu` to work is the `-l` option. The `-l` option stands for "line" and tells `cu` which port the modem is on. The command line that allows you to dial out is

   ```
   cu -s1200 -ltty0
   ```

   Enter this line and press RETURN. The word `Connected` should appear. 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`.

4. **The next sequence works only if you are using a Hayes-compatible modem, such as an Apple Personal Modem, but the principle is the same in all cases.**

   Once you are connected to the modem, there is a specific way of communicating with it and making it do what you want. If your modem is not Hayes compatible, your modem owner's manual will have information on how to communicate with it.

   Enter the command that tells your modem to dial a phone number. If your modem is Hayes compatible, try this:

   `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.

5. **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 in front of it. See `cu`(1C) in *A/UX Command Reference.*

6. **When you are ready to break the connection with the other computer, you must log out.**

7. **Once you are logged out, wait one second, then enter +++ and wait another second. Your modem answers OK on your screen.**

8. **Enter ATH.**

   This tells your modem to hang up. The modem responds with OK. If your modem is not Hayes compatible, the manual that comes with it describes how to disconnect from a remote computer.

9. **Enter the two-character sequence "tilde-dot":**

   This terminates the session with `cu`. The word `Disconnected` appears, and you are back at your own computer.

Dial-in access only

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 your system can use it as if he or she were connected directly to your computer via a terminal.

1. **To make your modem work as an incoming device, find this line in `/etc/inittab`:**

   ```
   00:2:off:/etc/getty tty0 mo_9600 # Port tty0 dialout
   ```

   and change it to

   ```
   00:2:respawn:/etc/getty tty0 mo_9600 # Port tty0 dialup
   ```

   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.

2. **To allow incoming calls, run `init q` to start the `getty`.**

   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.

Using `newconfig` to add a device requiring kernel modification

The `newconfig` program spares you the tedium and potential pitfalls of adding a new peripheral device by configuring the device driver and any software modules into the kernel for you. This program runs both the `newunix`(1M) and `autoconfig`(1M) programs automatically. For more complete information on these commands, refer to *A/UX System Administrator's Reference.*

- *Note:* You do not need to use `newconfig` to add terminals, printers, modems, Apple Tape Backup 40SC, CD-ROMs, 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.

1. **To add a new software module or driver to the kernel, run the `/etc/newconfig` program.**

   The `/etc/newconfig` program installs the files needed to add a software module to the kernel. You 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 slip
   ```

   installs the device driver for the Serial Line Interface Protocol. More than one device driver can be installed at a time; for example:

   ```
   /etc/newconfig nfs tc
   ```

   installs the Network File System and the tape controller driver.

2. **Use `newconfig` to create the new kernel containing the new software modules.**

The command

```
newconfig
```

describes the actions performed by `newconfig`, places a list of startup programs to run at boot time in the `/etc/startup` file, and creates a new kernel in `/unix`.

3. **Now you must reboot the new kernel before using the newly added device or the new software module.**

   Shut down the system and reboot the new kernel by following the directions given in Chapter 2, "System Startup and Shutdown."

`newconfig` and `newunix`

The `newconfig` program runs the `newunix` program to install the files for a particular software module into the directories required by `newconfig`. By default, `newconfig` builds a new kernel based on the information in `/etc/config/newunix`, the object modules in `/etc/boot.d`, and the files in `/etc/master.d`. Once you have added a new software module or driver to your system, it is always included when `autoconfig` builds a new kernel, unless you explicitly remove it.

- **To see which modules are currently configured in the kernel, enter the command**

  ```
  module_dump /unix
  ```

- **You can also use `/etc/newconfig` to remove software modules from the kernel. For example, to remove the software driver for the Apple Tape Backup 40SC, enter**

  ```
  /etc/newconfig notc
  ```

  - *Note:* The `tc` object module will be removed from the `/etc/boot.d` directory to create a new kernel that does not include the `tc` driver. The `newconfig` program builds a new kernel based on the object modules in `/etc/boot.d`. Because the `tc` module is no longer in this directory, the `tc` module is not included in the new kernel. To begin using the new kernel, shut down your system and reboot.

Adding new devices

When you add a new device, `/etc/newconfig` automatically runs the installation scripts provided with that device. An installation script typically installs files for that device in `/etc/install.d`. Follow the instructions provided with your device to install any software modules into the kernel. Typically, after you run the installation script for the new device, you will need to run `/etc/newconfig` and then reboot your system, as described in the previous section.

System V print spooler: `lp`

The following information is provided for backward compatibility with the System V print spooler.

△ **Important** Before using `lp`, you must turn off the `lpr` spooler daemon (`lpd`) and clear the `lpr` printer 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.
`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, a person or persons 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. Either the superuser or any user logged into the system as `lp` qualifies as the `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 *A/UX System Administrator's Reference.*

`lpsched` Routes user print requests to interface programs, which do the printing on devices. Also see `lpsched`(1M) in *A/UX System Administrator's Reference.*

`lpshut` Stops running `lpsched`. All printing activity is halted, but other `lp` commands may still be used. Also see `lpsched`(1M) in *A/UX System Administrator's Reference.*

`accept` Allows `lp` to accept output requests for destinations. Also see `accept`(1M) in *A/UX System Administrator's Reference.*

`reject` Prevents `lp` from accepting requests for particular destinations. Also see `reject`(1M) in *A/UX System Administrator's Reference.*

`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 *A/UX System Administrator's Reference.*

- *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 (`|`) or with the shell input redirect character (<). These two methods work because the user is feeding input to the `lp` command via the standard input. Here are two examples:

  ```
  lp < /etc/passwd
  pr abc | lp
  ```

Determining `lp` status

The `lpstat` command displays on the screen the status of printing requests, destinations, and the scheduler (`lpsched`). 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 (`lpsched`) 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 `lpsched` program routes the output requests (made with `lp`) through the appropriate printer interface programs to the printers. As noted previously, before a request can be printed, the scheduler (`lpsched`) must be running and the particular printer must be enabled and accepting requests.

Activating the scheduler

To activate `lpsched`, make sure the following line in the `/etc/rc` file is not "commented out" with number signs (#) at the start of the line:

```
rm -f /usr/spool/lp/SCHEDLOCK
```

Also be sure that the following line appears in `/etc/inittab:`

```
lp:2:once:/usr/lib/lpsched >/dev/syscon 2>&1
```

This starts the `lp` scheduler each time the system enters run level 2.

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.

△ **Important** Do not start the `lp` scheduler while `lpd` is running. △

◆ *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 by giving the commands

```
cp /dev/null /usr/spool/lp/oldlog
cp /dev/null /usr/spool/lp/log
```

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.

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 if `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.

- *Note:* Although it is possible to change the text files with a text editor, don't. 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 -p`*printer* `[-v`*device*`]` `[`*options*`]`
`lpadmin -x`*dest*
`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.

As you read, you should get a clear idea (perhaps in the form of a diagram you draw as you read) of which printer classes the examples establish, which printers belong to which classes, what models apply to what printers, and so on.

Introducing new destinations

You can add a new printer by giving the command
`lpadmin -p`*printer* `[-v`*device*`] [`*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 — Hard-wired printer or other file that is writable 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 (plugged directly into the computer). 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 *printer.* The *device* must be a pathname of a file that is writable 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 `cl1` and will use the same interface as printer `pr2`:**

  ```
  /usr/lib/lpadmin -ppr3 -v/dev/tty1 -epr2 -ccl1
  ```

Modifying existing destinations

You can use `lpadmin` to modify existing destinations. Always make modifications with respect to a printer name (`-p`*printer*). The form of the command is
`lpadmin -p`*printer 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 -ccl1
  ```
- **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 -ccl2
  ```

  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 `cl2`:**
  `/usr/lib/lpadmin -ppr1 -ccl2`

  The members of class `cl2` are now `pr2` and `pr1`, in that order. Requests routed to class `cl2` 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 `cl1`:**
  `/usr/lib/lpadmin -ppr2 -rcl1`
  `/usr/lib/lpadmin -ppr3 -rcl1`

  Because `pr3` was the last remaining member of class `cl1`, the class is removed.

- **Add `pr3` to a new class called `cl3`:**
  `/usr/lib/lpadmin -ppr3 -ccl3`

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` flag 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 `cl1` as the system default destination:**
  `/usr/lib/lpadmin -dcl1`

- **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 -x`*dest*

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 `cl2` is also removed because `pr2` was its only member.

- **Remove class `cl3`:**
  `/usr/lib/lpadmin -xcl3`

  Class `cl3` 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 `-d`*dest* 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 Diablo 1640 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, 12 is a command to the Diablo 1640 interface program to print output in 12-pitch mode. Other 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
  ```

- **Re-enable `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-`*nnn* to `bobby-`*nnn*. 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 the root user and 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 root user's mail for 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 -ulp 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 `lpsched`'s messages indicate damaged spooler configuration files (see "`lp` System Files," later in this chapter), use the `lpadmin` command to remake the `lp` system (see "Configuring the `lp` System," earlier in this chapter).
   - 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 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`.**

   Do not remove the directories themselves. Use the `rm ./*` command, but with caution! Use `pwd` to be sure you are in the directory you think you are in!

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, readable and writable 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 chapter.

`/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 set user ID (SUID) bit turned on (see `chmod`(1) and `chown`(1) in *A/UX Command Reference*. The `lpsched` scheduler may be owned by either root or `lp`.

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

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 BSD and 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 backward compatibility

Note that this information does not apply to the Macintosh file system. A description of how `fsck` works and how to use it follows.

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, a system administrator who knows the structure and functioning of the file system must resolve the problem. The goal of this chapter is to provide you with this knowledge.

Before you begin this chapter you should know how to use the basic UNIX commands `ls`, `rm`, `cp`, `mv`, and `cd`. You should also be able to bring the system to single-user or multi-user status. (See Chapter 2, "System Startup and Shutdown," for more information.)

Introduction to `fsck`

The file-system check program `fsck` locates and resolves inconsistencies within a file system. It is part of the normal booting 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 in `/etc/fstab`. If `fsck` detects errors during the automatic boot procedures, and you do not select Repair in the dialog box, or if a message that says the file system cannot be repaired is displayed, you must run `fsck` at the command line interface after you log in. Use `fsck` at any other time that you suspect inconsistencies within the file system, such as immediately following a 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, 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, and inconsistencies within the file system result. Fortunately, in most cases you can resolve these inconsistencies by using the A/UX `fsck` program.

The `fsck` program checks the location of files on disk and uses redundancies and known parameters to resolve inconsistencies. 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. 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.

It is important 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`. This section gives a brief overview of the relevant file and directory information.

Partitions, file systems, and hierarchies

Each A/UX **file system** is the complete set of data structures, commands, and subroutines used to manipulate data stored on a physical device. On the A/UX system, a physical storage device (usually a disk) is divided into logical sections called **partitions,** each of which may contain an A/UX file system. You can use the Apple HD SC Setup software to establish or remove partitions. For complete instructions, see Chapter 4 in *Setting Up Accounts and Peripherals for A/UX.*

A **file system** resides on a partition that contains the data structures (directories, files, and inodes, 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. This coincides with the collection of all files on currently mounted file systems; see `mount`(1M) in *A/UX System Administrator's Reference.* From the user's point of view, the directory hierarchy resembles an inverted tree, branching out from the root directory (`/`).

Bytes and blocks

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 512 bytes (characters). 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.** The number of physical blocks per logical block is file-system dependent. SVFS typically uses two physical blocks per logical block—that is, it is a 1-kilobyte file system. In comparison, the UFS ratio is typically eight to 16 physical blocks for each logical block. UFS also supports a fragmented logical block to prevent disk waste. Each character in a file is represented by a byte of information, and each byte resides in a block.

A file smaller than one logical block, called a **data block,** resides in one location on the disk. A file larger than this resides in different portions of the disk. Ideally, the data blocks of a file should be as near to one another as possible to optimize disk I/O operations. However, in reality, as the disk is used and files are deleted, data blocks become increasingly scattered. To handle this, each file's inode points to each block of the file in sequence.

To find out how many blocks each of the files in your current directory occupies, enter the command

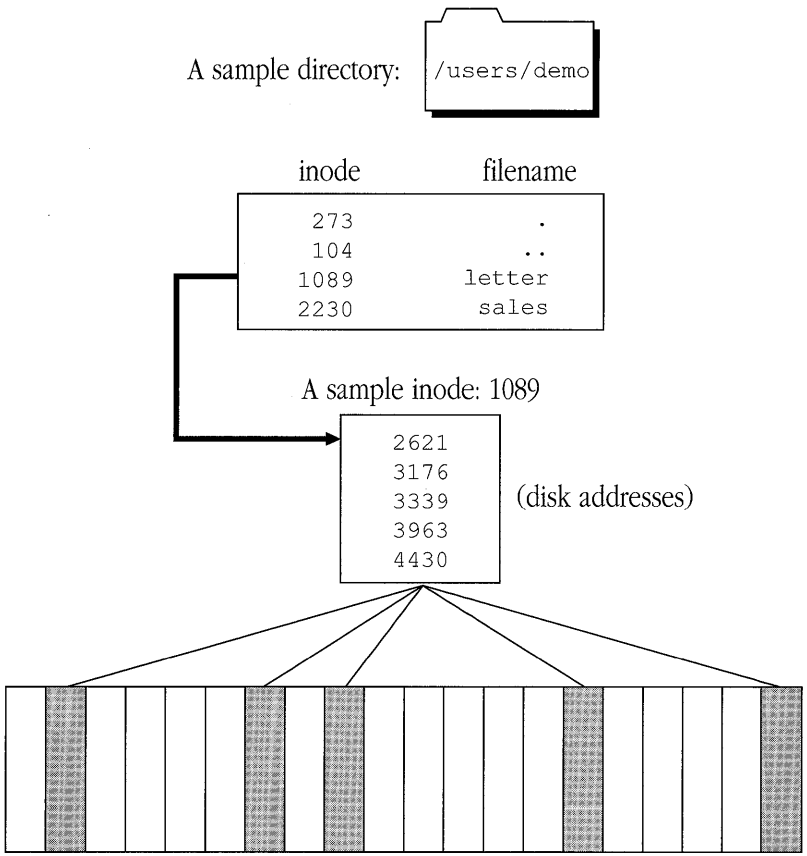
```
ls -s
```

The number that appears next to each filename is a count of physical blocks used by that file. There should be one block for each 512 characters and an additional block for any remaining characters. For example, a file with 512 characters occupies one physical block, and a file with 513 characters occupies two physical blocks. Also, you must round up to the logical block size—that is, if the file system's logical block size is 1 kilobyte, then the smallest file takes up two physical blocks. In the example, 512 and 513 would both occupy two physical blocks.

Inodes

Inodes contain information about files, the most important of which is a list of locations on the disk where the 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"). Figure 8-1 illustrates the relationship between the directory `/users/demo`, a file in that 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



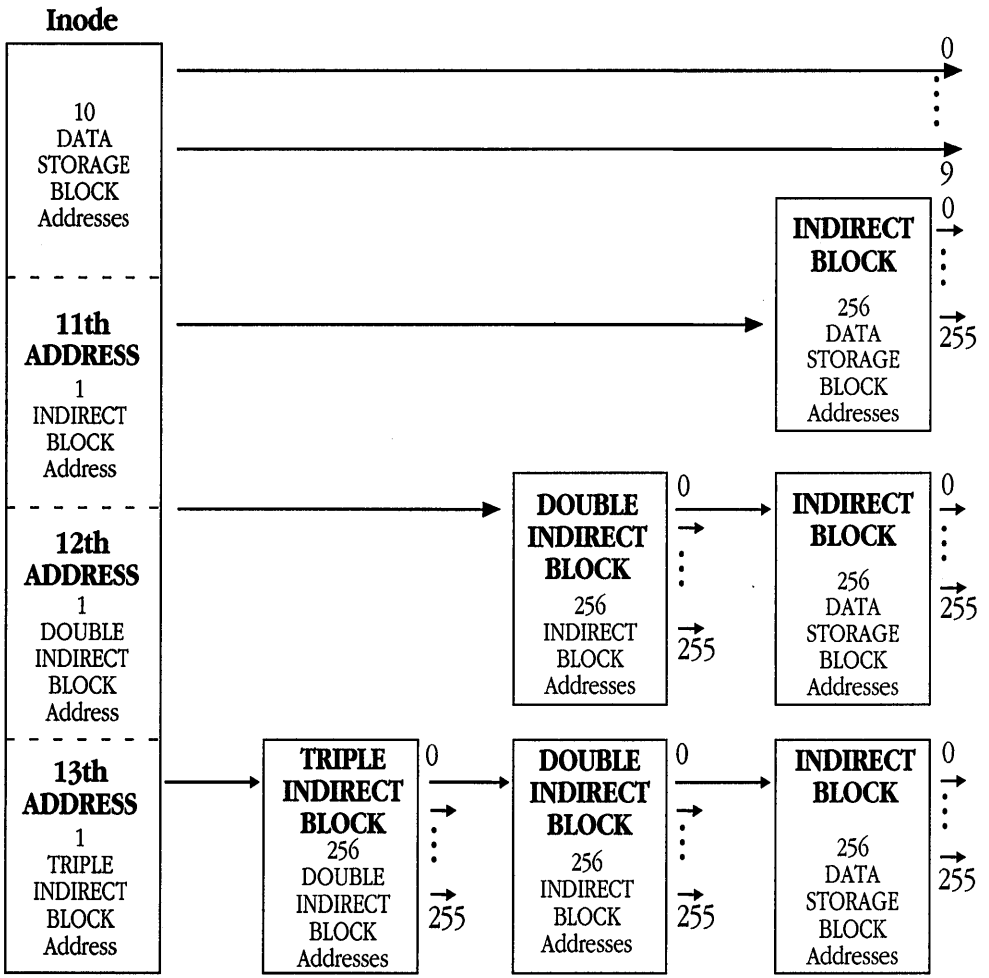
Disk blocks occupied by the file `letter`

Direct and indirect blocks

The inode contains 13 disk addresses. The first 10 addresses point to the first 10 physical 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 10 addresses have been used.

If a file has used up the direct data blocks, the 11th address given in the inode is then taken into consideration. The 11th disk address points to an **indirect block.** An indirect block contains the addresses for the next 256 logical blocks that the file can use. Figure 8-2 illustrates these connections.

■ **Figure 8-2** Indirect blocks



If the file is larger than 266 blocks (10 blocks for the first 10 addresses, 256 for the 11th), it continues at the 12th address given in the inode. The 12th address points to a **double indirect block,** which refers to up to 256 indirect blocks that the file can use. This gives a total of 65,802 logical blocks of data: 10 from the first 10 addresses plus 256 from the indirect block plus 65,536 from the double indirect block. For a 1-kilobyte file system, a file that uses the double indirect block may hold 64 kilobytes. On an 8-kilobyte file system, the same file holds over 512 kilobytes.

If the file is even larger, the 13th address in the inode is called into action. This last address points to a **triple indirect block,** which refers to up to 256 double indirect blocks, each of which in turn refers to up to 256 indirect blocks, and so on. Now you have logical blocks

(10 + 256) + (256 * 256) + (256 * 256 * 256)

This is the maximum size a file can be in a file system that has 1024-byte logical blocks. A system with a larger bytes-per-block value could have a much larger theoretical limit. But because the file size is represented in an inode by a signed 32-bit quantity, a file can never get larger, in practice, than about 2 gigabytes.

More on inodes

Inodes contain information about the location of the data blocks that make up a file. Figure 8-3 illustrates the other important information that inodes contain about a file.

As shown in Figure 8-3, 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. For further information, see `ls`(1) in *A/UX Command Reference.*

■ **Figure 8-3** Additional information in an inode

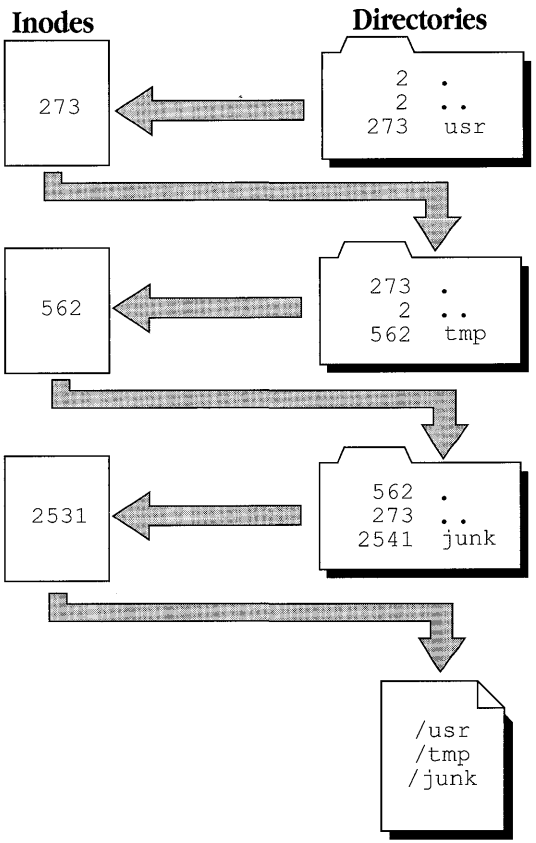
A sample inode: 1089

666	Permissions
1	Number of links
102	Owner
400	Group
5432	Size
588019673	Time last accessed
588019673	Time last modified
588019673	Inode last modified
2621	Disk address list
3176	
3339	
3963	
4430	

Starting from the top

Figure 8-4 illustrates the connection, through multiple inodes, between the root directory and a file located several levels below the root directory.

■ **Figure 8-4** File-directory connection through inodes



Inode location

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.

Superblock

Each A/UX file system is described by its **superblock,** which is located at the beginning of the file system's disk partition. 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 to identify 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 shut down 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 information about 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 writes the modified versions back into secondary memory for future use.

When a program reads data from or writes data to a tape or a disk, 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 to do physical 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 **data 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 writes refers generally to unusual bad writes to the buffer, and bad disk writes 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/rdsk/c0d0s0 and /dev/dsk/c0d0s0 by giving the command

```
ls -l /dev/rdsk/c0d0s0 /dev/dsk/c0d0s0
```

In response, the system displays

```
crw------- 2 bin 24, 0 May 25 1990 /dev/rdsk/c0d0s0
brw------- 2 bin 24, 0 May 25 1990 /dev/dsk/c0d0s0
```

The first character in the permissions field of /dev/rdsk/c0d0s0 is c, and the first character in the permissions field of /dev/dsk/c0d0s0 is b. 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 1990 /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 earlier section, "Special Files and the /dev Directory."

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, tape, 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.

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 the disk device).

Devices (and therefore device drivers and their corresponding special files) come in two forms, `b` (block) and `c` (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, one block at a time. **Character devices** such as terminals, line printers, and modems are written to, and read from, one character at a time. Another name for character devices is **raw devices.**

Each 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`, `volcopy`, and `fsck`, run faster when accessing the disk as a character device. For example,

```
fsck /dev/rdsk/c0d0s0
```

is faster than

```
fsck /dev/dsk/c0d0s0
```

The listing

```
crw------- 1 bin 24, 0 May 25 1990 /dev/rdsk/c0d0s0
brw------- 1 bin 24, 0 May 25 1990 /dev/dsk/c0d0s0
```

is clearly a listing of a character (raw) device and a block device. The first has a `c` and the second a `b` as the first entry in the permissions field. Thus the same device can have two different interfaces.

How `fsck` works

As you open, create, and modify files, the system keeps track of all pertinent information about them. This 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 in main memory. If the system crashes or becomes corrupt for any reason, the various file systems will probably become inconsistent. This inconsistency arises because the information in main memory was not written to disk before the problem, whereas other information is written immediately.

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 is due to an inconsistent update or one that was performed out of order. To understand the problems `fsck` is designed to solve, you need 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

Contains information about the size of the file system and inode list, part of the free list, a count of free blocks, a count of free inodes, and part of the free inode list.

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) 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

Contains information about the inode's type (which may be directory, data, or special), the number of directory entries linked to it, the list of blocks claimed by the inode, and the inode's size. An inode is written to disk when the file associated with it is closed. In fact, all **in-core blocks** 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.

Indirect blocks

Can be one of three types: single-indirect, double-indirect, or triple-indirect.

Indirect blocks, as well as the first ten 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` command is issued.

Inconsistencies in an indirect block directly affect the inode that owns it.

You can check the following inconsistencies: blocks already claimed by another inode and block numbers outside the range of the file system.

Data block

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

If a directory entry inode number (that is, a file's inode) points to an unallocated inode, `fsck` may remove that directory entry, depending on how `fsck` was invoked. (For instance, it removes the entry if invoked with the `y` flag option.) See "`fsck` Messages," later in this chapter. This condition may occur when the data blocks containing the directory entries are modified and written out while the inode is not yet written out.

If a directory entry inode number points beyond the end of the inode list, `fsck` may remove that directory entry. This condition occurs if bad data is written into a directory data block.

The directory inode number entry for the dot file `.` should be the first entry in the directory data block. Its value should be equal to the inode number for the directory data block.

The directory inode number entry for the dot file `..` should be the second entry in the directory data block. Its value should be equal to the inode number for the parent of the directory entry (or the inode number of the directory data block if the directory is the root directory).

If the directory inode numbers are incorrect, `fsck` may replace them with the correct values.

The `fsck` program checks the general connectivity of the file system. If directories are found not to be linked into the file system, `fsck` links them back into the file system's `lost+found` directory. This condition can be caused if inodes are written to disk but the corresponding directory data blocks are not.

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 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.

You can check the following inconsistencies: a list count outside of range, block numbers outside of range, and blocks already associated with the file system.

`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.

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 UFS: Check cylinder groups

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 SVFS: Check free list

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 programmed into the system startup procedure (see Chapter 2, "System Startup and Shutdown") and is automatically run each time you start up A/UX.

You can also invoke `fsck` at any time during a session by bringing the system down to single-user mode and entering

`fsck` [*options*] [*file-systems*]

You can specify *options* to direct `fsck` to run in a different way; see `fsck`(1M) in *A/UX System Administrator's Reference* for a complete list of options. You can specify *file-systems* to run `fsck` on a different file system. File system names are defined in the file `/etc/fstab`; see `fstab`(4) in *A/UX Programmer's Reference* for details. If you enter `fsck` without options or file system names, it will run on all file systems.

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 such as

```
/dev/dsk/c0d0s0 POSSIBLE FILE SIZE ERROR I=2405
```

The message can also look like this:

```
/dev/dsk/c0d0s0 FREE INODE COUNT WRONG IN SUPERBLK FIX?
```

The second message illustrates one of `fsck`'s interactive error messages. 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.

fsck options

You can specify the following flag options on the `fsck` command line for both file systems (SVFS and UFS file-system-specific options follow):

`-y` Automatic yes. The `y` flag option automatically provides a yes response to all questions that `fsck` asks.

`-n` Automatic no. The `n` flag option automatically provides a no response to all questions that `fsck` asks. The file system is not opened for writing.

`-mtimeout` Tells `fsck` to use a Macintosh interface. `fsck` directs StartMonitor to move the progress bar during the boot sequence. If `fsck` finds a problem with the file system, the option posts a Macintosh alert (dialog box) asking if the user wants to repair the file system. If the user selects the default Repair button, `fsck` assumes a yes reponse to all further questions regarding the file system.

If a timeout value greater than 0 is given, the dialog box automatically chooses the default button after that number of seconds. The timeout default is 0, which indicates that the alert should not time out automatically.

`-p` [*pass-to-start*] Automatic fix. If you give the `p` flag option, `fsck` automatically fixes those file system inconsistencies that it can fix without your assistance. The optional pass-to-start argument specifies the starting pass number. You cannot give `fsck` a starting pass number except as an argument to `p`. The starting pass number limits which file systems `fsck` checks; `fsck` looks in the `/etc/fstab` file and checks those file systems with pass numbers equal to or greater than the pass-to-start argument. If you don't specify a pass number, the default is 1.

`-q` Quiet `fsck`. If you give the `q` option, `fsck` does not print size-check messages in Phase 1. Unreferenced FIFOs are silently removed. If `fsck` requires it, counts in the superblock are automatically fixed, and the free list is salvaged.

SVFS-specific options

The following options apply only to the SVFS file system:

`-f` Fast check. This causes `fsck` to check blocks and sizes (Phase 1) and check the free list (Phase 5). The free list is reconstructed (Phase 6) if necessary.

`-s` *x* Unconditional reconstruction of the free list. The `s` option causes `fsck` to ignore the actual free list and (unconditionally) reconstruct a new one by rewriting the superblock of the file system. The file system should be unmounted while this is done; if this is not possible, you should be careful that the system is quiescent and that it is rebooted immediately afterward. This precaution is necessary so that the old, bad in-core copy of the superblock will not continue to be used or written on the file system.

The `s`*x* flag option allows you to create an optimal free-list organization. The option has the following form:

`-s` *blocks-per-cylinder*`:` *blocks-to-skip*

If *x* is not given, `fsck` uses the values specified when the file system was created. If these values were not specified, then the value `400:7` is used.

`-S` *x* Conditional reconstruction of the free list. This flag option is like `s`*x*, except that the free list is rebuilt only if no discrepancies were discovered in the file system. Using `S` forces a no response to all questions that `fsck` asks. This flag option is useful for forcing free-list reorganization on uncontaminated file systems.

`-t` *file* Named scratch file. If `fsck` cannot obtain enough memory to keep its tables, it uses a scratch file. If the `t` option is specified, the file named in the next argument is used as the scratch file, if needed. Without the `t` flag, `fsck` prompts the operator for the name of the scratch file. The file chosen should not be on the file system being checked. If the scratch file is not a special file or did not already exist, `fsck` removes it at the end of the run.

`-D [options]` Options. If the *options* argument is missing, `fsck` merely checks directories for bad blocks. The *options* argument may be any of the following:

`B` Check for and clear parity bits in filenames.

`C` Check whether all trailing characters in the filename are null.

`CZ` Check and write nulls in all trailing characters in the filename.

UFS-specific options

The following option applies only to the UFS file system:

`-b` *block-number* Use the block specified immediately after the flag as the superblock for the file system. Block 32 is always an alternative superblock. See the `/etc/fstab` file for a default list of file systems to check.

`fsck`: a sample interaction

If you bring the system down to single-user mode and enter

```
fsck -n
```

`fsck` starts running. Because you didn't specify a file system, `fsck` reads the file `/etc/fstab`, which contains a list of the files to be checked in this case. 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, you are safe.

◆ *Note:* Unless you know exactly what you're doing, always invoke `fsck` a 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 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 gorups
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`."

The preceding example illustrates a routine check during which no problems or inconsistencies were found. If `fsck` finds a problem, you will see a screen message similar to this one:

```
/dev/rdsk/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` Messages," later in this chapter.

You can also invoke `fsck` with the `s`*x* or `S`*x* flag options discussed earlier. The `s`*x* flag option unconditionally makes a new free list, discarding the present one; the `S`*x* flag option does the same only if nothing is wrong in the free list. The immediate result of these options is to speed up disk operations in busy file systems, because a new free list is likely to have more contiguous portions of free space.

The `D` flag option is useful for doing extra consistency checks on directories, and it does not prolong the process significantly. It is included in your startup `fsck` procedures. For more information about these and other options, see `fsck`(1M) in *A/UX System Administrator's Reference.*

Multiple file systems and `fsck`

File-system checks occur when the system goes from single- to multi-user mode. You need to take a few steps to make sure that `fsck` automatically checks file systems other than the root file system when you respond yes to the prompt "Do you want to check the file systems?"

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 `/etc/fstab`. The determinant fields in the `/etc/fstab` file are the *type* of the file system and the *pass-number.* Figure 8-5 shows how `fsck` uses its options and these two fields to decide whether to check a file system.

■ **Figure 8-5** How `fsck` decides whether to check a file system

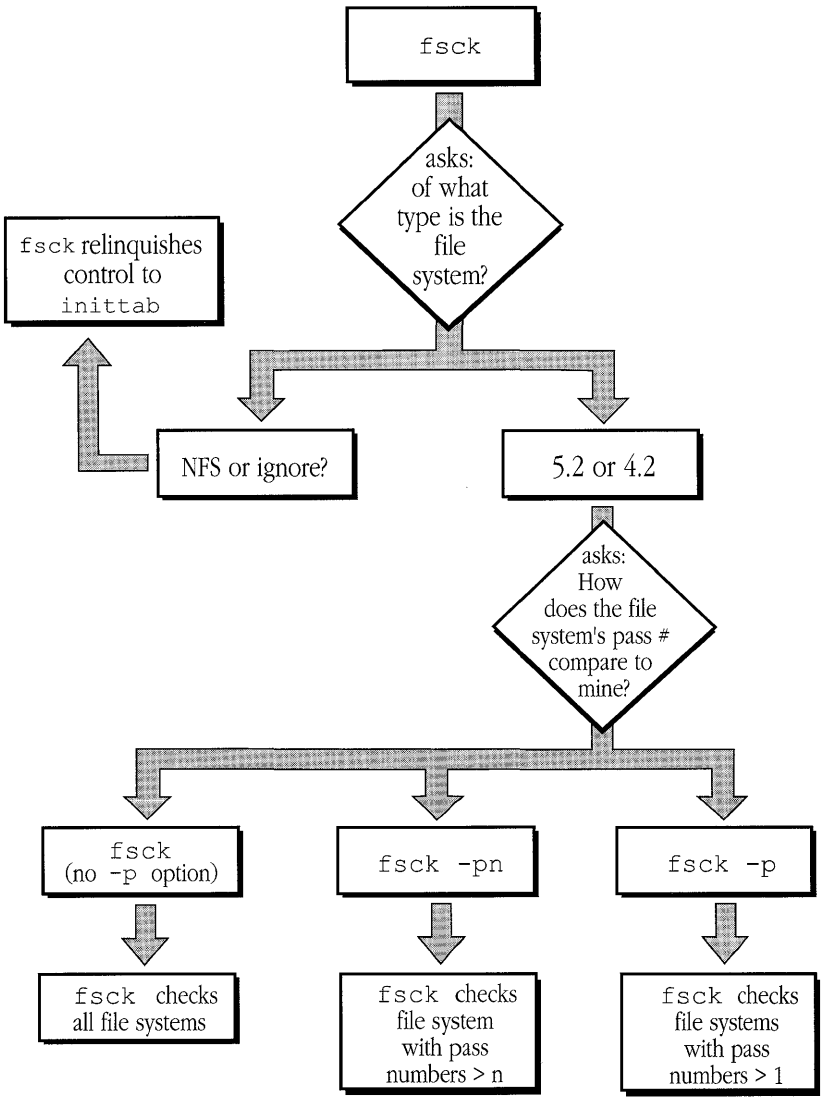
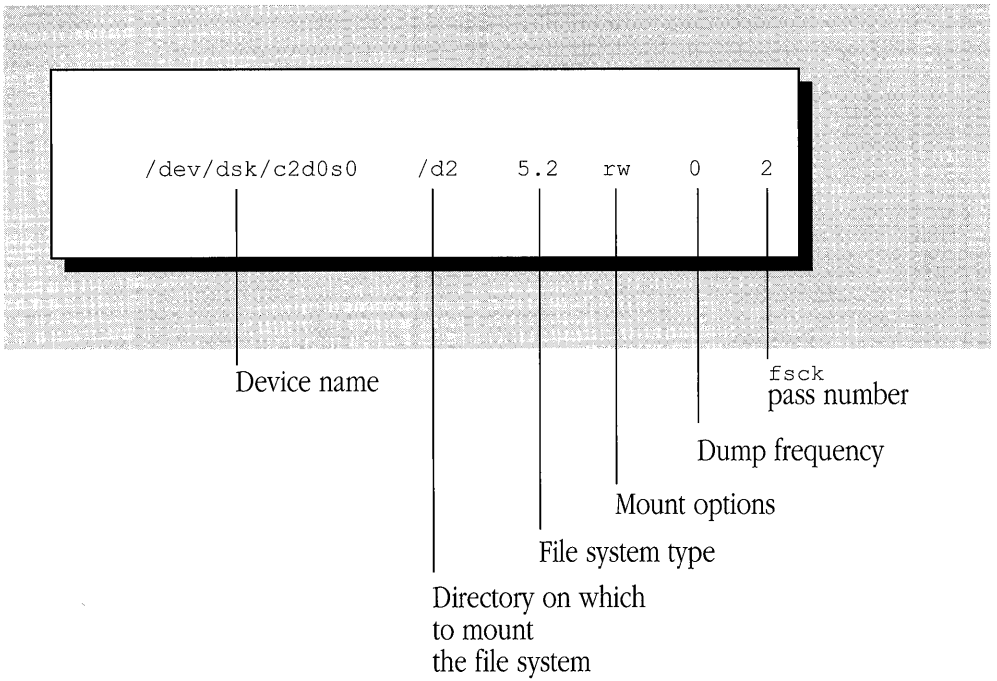


Figure 8-6 shows a sample annotated `/etc/fstab` file. The values it contains cause `fsck` to check the listed file system.

■ **Figure 8-6** A description of sample entries in `/etc/fstab`



1. **Open the `/etc/fstab` file and in the rightmost field set the *pass-number* for the file system to a number greater than or equal to 2.**
2. **Set the *type* of the file system to 4.2 or 5.2 (not ignore).**

For more about the pass number, see the `p` option in "`fsck` Options," earlier in this chapter.

Once you set these fields in the `/etc/fstab` file, you can have `fsck` check all your file systems on command by entering

```
fsck -p
```

Or you can run `fsck` on an individual file system by giving the command

```
fsck file-systems
```

fsck messages

The fsck program is a multi-pass 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.

Normally, fsck is run noninteractively to correct the file systems after an unclean halt. These actions are a proper subset of those that fsck takes when it is running interactively. Note that many errors have several options that the operator can take.

When an inconsistency is detected, fsck reports the error condition to the operator 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 responses, and the related error conditions. The messages are organized by the fsck phase in which they can occur. The error conditions are organized by the phase of the fsck program in which they can occur. The error conditions that may occur in more than one phase are discussed in the following section, "fsck Initialization Phase Messages: UFS-specific."

fsck initialization phase messages: UFS-specific

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.

fsck option errors

The following messages may appear when you specify the command line incorrectly. See fsck(1M) in *A/UX System Administrator's Reference* for further details.

`c` option? — *c* stands for any character that is not a legal flag option to fsck. Legal options are b, y, n, and p; fsck terminates on this error condition.

Memory request errors

The following messages mean that `fsck`'s request for memory for its virtual memory tables failed. As a consequence, `fsck` terminates. This indicates a serious problem that requires technical assistance.

```
cannot alloc NNN bytes for blockmap
cannot alloc NNN bytes for freemap
cannot alloc NNN bytes for statemap
cannot alloc NNN bytes for lncntp
```

Errors in opening files

The following message may appear when `fsck` cannot open a file or file system.

`can't open checklist file:` *f*
: The default file-system check file *f* (usually `/etc/fstab`) cannot be opened for reading; `fsck` terminates on this error condition. Check the access modes of *f* and modify accordingly.

`can't open` *f*
: The file system *f* cannot be opened for reading. The `fsck` program ignores this file system and continues checking the next file system given. Check the access modes of *f* and modify accordingly.

f `is not a block or character device;OK?`
: The `fsck` program has been given a regular filename by mistake. Check the type of the file specified.

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.

File status errors

The following messages may appear when `fsck` cannot obtain a file's status. These errors may indicate a serious problem that you may not be able to solve without technical assistance.

`can't stat root`
: The `fsck` program's request for statistics about the root directory (`/`) failed; `fsck` terminates on this error condition.

`can't stat` *f*
`can't make sense out of name` *f*
: The `fsck` program request for statistics about the file system *f* failed. It ignores this file system and continues checking the next file system given. Check the access modes of *f*.

f: `(NO WRITE)`
: Either the `-n` flag was specified or `fsck`'s attempt to open the file system *f* for writing failed. When running manually, all the diagnostics are printed out, but no modifications are attempted to fix them.

Superblock errors

`UNDEFINED OPTIMIZATION IN SUPERBLOCK`
`SET TO DEFAULT?`
: The superblock optimization parameter is neither `OPT_TIME` nor `OPT_SPACE`.

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.

`IMPOSSIBLE MINFREE=`*d* `IN SUPERBLOCK`
`SET TO DEFAULT?`

The superblock minimum space percentage is greater than 99 percent or less than 0 percent.

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 32 kilobytes, specifying `-b 32` is a good first choice.

Interactive messages

These messages require your yes or no response to the `CONTINUE?` prompt. "Yes" may be `Y` or `y`, and "no" may be `N` or `n`; these responses are shown in uppercase here. The following messages may indicate a serious problem because the file system cannot be completely checked. You may need to obtain additional technical assistance.

`CANNOT SEEK: BLK` *b*
`CONTINUE?` 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. If the block *b* was part of the virtual memory buffer cache, `fsck` stops with the message `Fatal I/O error`.

N Stops `fsck`.

`CANNOT READ: BLK` *b*

`CONTINUE?` The `fsck` program request for reading 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 and 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. If the disk is experiencing hardware problems, the problem will persist. This error condition will not allow a complete check of the file system. Rerun `fsck` to recheck this file system. If the block was part of the virtual memory buffer cache, `fsck` ends the program with the message `Fatal I/O error`.

N Stops `fsck`.

`CANNOT WRITE: BLK` *b*

`CONTINUE?` The `fsck` program's request for writing a specified block number *b* in the file system failed. The disk is write-protected.

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. Run `fsck` a second time to recheck this file system. If the block was part of the virtual memory buffer cache, `fsck` terminates with the message `Fatal I/O error`.

N Stops `fsck`.

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.

Inode type errors

Each inode contains a mode word that describes the type and state of the inode.

Inodes may be one of five types: regular, directory, special block, special character, or FIFO, according to the file involved. If an inode is not one of these types, it is of an illegal type. If this is the case, tell `fsck` to clear the inode.

`UNKNOWN FILE TYPE I=`*i*

`CLEAR?` The mode word of the inode *i* indicates that the inode is not a special block, special character, socket, regular, symbolic link, or directory inode.

Possible responses to the `CLEAR?` prompt are

Y Deallocates inode *i* by zeroing its contents. This action always invokes the `UNALLOCATED` error condition in Phase 2 for each directory entry pointing to this inode.

N Ignores this error condition.

`PARTIALLY TRUNCATED INODE I=`*i*

`SALVAGE?` 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.

Possible responses to `SALVAGE?` prompt are

Y Completes the truncation to the size specified in the inode.

N Ignores this error condition.

Zero-link-count table errors

Each inode contains a stored count of the total number of directory entries linked to the inode. The `fsck` program verifies the link count of each inode by traversing down the total directory structure, starting from the root directory, and calculating an actual link count for each inode.

If the stored link count is nonzero and the actual link count is zero, no directory entry appears for the inode. If the stored and actual link counts are nonzero and unequal, a directory entry may have been added or removed without the inode being updated.

If the stored link count is nonzero and the actual link count is zero, `fsck` can link the disconnected file to the `lost+found` directory (at your direction). If the stored and actual link counts are nonzero and unequal, `fsck` can replace the stored link count with the actual link count.

`LINK COUNT TABLE OVERFLOW`
`CONTINUE?` An internal table for `fsck` containing allocated inodes with a link count of zero cannot allocate more memory. Increase the virtual memory for `fsck`.

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. 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 the program.

Bad or duplicate blocks

Each inode contains a list of pointers to lists (indirect blocks) of all the blocks claimed by the inode.

The `fsck` program checks each block number claimed by an inode for a value lower than that of the first data block or greater than that of the last block in the file system. If the block number is outside this range, it is a bad block number. If an indirect block was not written to disk, an inode may contain many bad blocks. In this case, `fsck` clears both inodes (at your direction).

The `fsck` program compares each block number claimed by an inode to a list of already allocated blocks. If a block number is already claimed by another inode, the block number is added to a list of duplicate blocks. Otherwise, the list of allocated blocks is updated to include the block number. If there are any duplicate blocks, `fsck` makes a partial second pass of the inode list to find the inode of the duplicated block. This is necessary because without examining the files associated with these inodes for correct content, `fsck` does not have enough information to decide which inode is corrupted and should be cleared. Usually, the inode with the earliest modify time is incorrect and should be cleared. This condition may be the result of a file system containing blocks claimed by both the free list and other parts of the file system.

A large number of duplicate blocks in an inode may be due to an indirect block not being written to disk. In this case, `fsck` clears both inodes (at your direction).

`BAD I=`*i* Inode *i* contains block number *b* with a number lower than the number of the first data block in the file system or greater than the number of the last block in the file system. This error condition may 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.

`EXCESSIVE BAD BLKS I=`*i*
`CONTINUE?` 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 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 recheck this file system.

N Stops the program.

`BAD STATE` *ddd* `TO BLKERR`
An internal error has scrambled `fsck`'s state map to have the impossible value *ddd*. The `fsck` program exits immediately. Seek technical assistance.

b DUP I=*i*
: Inode *i* contains block number *b* that 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 action always invokes Phase 1B and the BAD/DUP error condition in Phase 2 and Phase 4.

EXCESSIVE DUP BLKS I=*i*
CONTINUE?
: 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.

DUP TABLE OVERFLOW
CONTINUE?
: An internal table in `fsck` containing duplicate block numbers cannot allocate any more space. Increase the amount of virtual memory available to `fsck`.

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 another duplicate block is found, this error condition is repeated.

N
: Stops the program.

Inode format errors

Each inode contains a mode word that describes the type and state of the inode. Inodes may be found in one of three states: allocated, unallocated, and neither allocated nor unallocated. This last state indicates an incorrectly formatted inode. An inode can get in this state if bad data is written into the inode list, a possible result of a hardware failure. The only corrective action is for `fsck` to deallocate the inode.

`PARTIALLY ALLOCATED INODE I=`*i*
`CLEAR?` Inode *i* is neither allocated nor unallocated.

Possible responses to the `CLEAR?` prompt are

Y Deallocates inode *i* by zeroing its contents.

N Ignores this error condition.

`INCORRECT BLOCK COUNT I=`*i* (*x* should be *y*)
`CORRECT?` The block count for inode *i* is *x* blocks, but should be *y* blocks.

Possible responses to the `CORRECT?` prompt are

Y Replaces the block count of inode *i* with *y*.

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* Inode *i* contains block number *b*, which is already claimed by another inode. This error condition action 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.

Root inode mode and status errors

`ROOT INODE UNALLOCATED`
`ALLOCATE?` The root inode (usually inode number 2) has no allocate mode bits. This should never happen.

Possible responses to the `ALLOCATE?` prompt are

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.`

N Causes `fsck` to exit.

`ROOT INODE NOT DIRECTORY`
`REALLOCATE?` The root inode (usually inode number 2) is not of the directory type.

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 `lost+found`. If the attempt to allocate the root inode fails, `fsck` exits with the message `CANNOT ALLOCATE ROOT INODE.`

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.

`DUPS/BAD IN ROOT INODE`
`REALLOCATE?` Phase 1 or Phase 1B has found duplicate blocks or bad blocks in the root inode (usually inode number 2) for the file system.

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.`

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.

NAME TOO LONG *f*

An excessively long pathname has been found. This usually indicates loops in the file-system name space. This can occur if the superuser has made circular links to directories. The offending links must be removed by a technical expert.

Directory inode pointers range errors

i OUT OF RANGE I=*i* NAME=*f*
REMOVE? 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*.

N Ignores this error condition.

Directory entries pointing to bad inodes

UNALLOCATED I = *i* OWNER=*o* MODE=*m* SIZE=*s* MTIME=*t* TYPE=*f*
REMOVE? A directory or file entry *f* points to an unallocated inode *i*. The owner *o*, mode *m*, size *s*, modify time *t*, and type *f* are printed.

Possible responses to the REMOVE? prompt are

Y Removes the directory entry *f*.

N Ignores this error condition.

`DUP/BAD I=`*i* `OWNER=`*o* `MODE=`*m* `SIZE=`*s* `MTIME=`*t* `TYPE=`*f*

`REMOVE?` Phase 1 or Phase 1B has found duplicate blocks or bad blocks associated with directory or file entry *f* inode. The owner *o*, mode *m*, size *s*, modify time *t*, and directory type *f* are printed.

Possible responses to the `REMOVE?` prompt are

Y Removes the directory entry *f*.

N Ignores this error condition.

`ZERO LENGTH DIRECTORY I=`*i* `OWNER=`*o* `MODE=`*m* `SIZE=`*s* `MTIME=`*t* `DIR=`*f*

`REMOVE?` A directory entry *f* has a size *s* that is zero. The owner *o*, mode *m*, size *s*, modify time *t*, and directory name *f* are printed.

Possible responses to the `REMOVE?` prompt are

Y Removes the directory entry *f*; this action always invokes the `BAD/DUP` error condition in Phase 4.

N Ignores this error condition.

`DIRECTORY TOO SHORT I=` *i* `OWNER=`*o* `MODE=`*m* `SIZE=`*s* `MTIME=`*t* `DIR=`*f*

`FIX?` A directory *f* has been found whose size *s* is less than the minimum size directory. The owner *o*, mode *m*, size *s*, modify time *t*, and directory name *f* are printed.

Possible responses to the `FIX?` prompt are

Y Increases the size of the directory to the minimum directory size.

N Ignores this directory.

`DIRECTORY` *f* `LENGTH` *s* `NOT MULTIPLE OF` *b*

`ADJUST?` A directory *f* has been found with size *s* that is not a multiple of the directory blocksize.

Possible responses to the `ADJUST?` prompt are

Y The length is rounded up to the appropriate block size. This error can occur on UFS file systems.

N Ignores this error condition.

`DIRECTORY CORRUPTED I=`*i* `OWNER=`*o* `MODE=`*m* `SIZE=`*s* `MTIME=`*t* `DIR=`*f*

`SALVAGE?` A directory with an inconsistent internal state has been found. Possible responses to the `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, and should be taken only after other recovery efforts have failed.

N Skips up to the next directory boundary and resumes reading, but does not modify the directory.

`BAD INODE NUMBER FOR '.' I=`*i* `OWNER=`*o* `MODE=`*m* `SIZE=`*s* `MTIME=`*t* `DIR=`*f*

`FIX?` A directory *i* has been found whose inode number for `'.'` does not equal *i*.

Possible responses to the `FIX?` prompt are

Y Changes the inode number for `'.'` to be equal to *i*.

N Leaves the inode number for `'.'` unchanged.

`MISSING '.' I=` *i* `OWNER=`*o* `MODE=`*m* `SIZE=`*s* `MTIME=`*t* `DIR=`*f*

`FIX?` A directory *i* has been found whose first entry is unallocated.

Possible responses to the `FIX?` prompt are

Y Builds an entry for `'.'` with inode number equal to *i*.

N Leaves the directory unchanged.

`MISSING '.' I=OWNER=`*o* `MODE=`*m* `SIZE=`*s* `MTIME=`*t* `DIR=`*f*

`CANNOT FIX, FIRST ENTRY IN DIRECTORY CONTAINS`*f*

A directory *i* has been found whose first entry is *f*. The `fsck` program cannot resolve this problem. The file system should be mounted and the offending entry *f* moved elsewhere. The file system should then be unmounted and `fsck` should be run again.

`MISSING '.' I=` *i* `OWNER=`*o* `MODE=`*m* `SIZE=`*s* `MTIME=`*t* `DIR=`*f*

`CANNOT FIX, INSUFFICIENT SPACE TO ADD '.'`

A directory *i* has been found whose first entry is not `'.'`. The `fsck` program cannot resolve this problem. Seek technical assistance.

`EXTRA '.' ENTRY I=`*i* `OWNER=` *o* `MODE=`*m* `SIZE=`*s* `MTIME=`*t* `DIR =`*f*

`FIX?` A directory *i* has been found that has more than one entry for `'.'`.

Possible responses to the `FIX?` prompt are

Y Removes the extra entry for `'.'`.

N Leaves the directory unchanged.

`BAD INODE NUMBER FOR '..' I=`*i* `OWNER=`*o* `MODE=`*m* `SIZE=`*s* `MTIME=`*t* `DIR =`*f*
`FIX?` A directory has been found whose inode number for `'..'` does not equal the parent of *i*.

Possible responses to the `FIX?` prompt are

Y Changes the inode number for `'..'` to be equal to the parent of (`".."`) in the root inode points to itself).

N Leaves the inode number for `'..'` unchanged.

`MISSING '..' I=`*i* `OWNER=`*o* `MODE=`*m* `SIZE=`*s* `MTIME=`*t* `DIR =`*f*
`FIX?` A directory *i* has been found whose first entry is unallocated.

Possible responses to the `FIX?` prompt are

Y Builds an entry for `'..'` with inode number equal to the parent of (`".."` in the root inode 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*
A directory *i* has been found whose second entry is *f*. The `fsck` program cannot resolve this problem. The file system should be mounted and the offending entry *f* moved elsewhere. The file system should then be unmounted and `fsck` should be run again.

`MISSING '..' I= `*i* `OWNER=`*o* `MODE=`*m* `SIZE=`*s* `MTIME=`*t* `DIR=`*f*
`CANNOT FIX, INSUFFICIENT SPACE TO ADD '..'`
A directory *i* has been found whose second entry is not `'..'`. The `fsck` program cannot resolve this problem. Seek technical assistance.

`EXTRA '..' ENTRY I =`*i* `OWNER=`*o* `MODE=`*m* `SIZE=`*s* `MTIME=`*t* `DIR =`*f*
`FIX?` A directory *i* has been found that has more than one entry for `'..'`.

Possible responses to the `FIX?` prompt are

Y Removes the extra entry for `'..'`.

N Leaves the directory unchanged.

n IS AN EXTRANEOUS HARD LINK TO A DIRECTORY *d*
REMOVE? The `fsck` program has found a hard link, *n*, to a directory, *d*.

Possible responses to the REMOVE? prompt are

Y Deletes the extraneous entry, *n*.

N Ignores the error condition.

BAD INODE *S* TO DESCEND
An internal error has caused an impossible state *s* to be passed to the routine that descends the file-system directory structure. The `fsck` program exits. See a technical expert.

BAD RETURN STATE S FROM DESCEND
An internal error has caused an impossible state *s* to be returned from the routine that descends the file-system directory structure. The `fsck` program exits. See a technical expert.

BAD STATE *s* FOR ROOT INODE
An internal error has caused an impossible state *s* to be assigned to the root inode. The `fsck` program exits. See a technical expert.

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.

UNREF DIR I= *i* OWNER=*o* MODE=*m* SIZE=*s* MTIME=*t*
RECONNECT? The directory inode *i* was not connected to a directory entry when the file system was traversed. The owner *o*, mode *m*, size *s*, and modify time *t* of 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.

lost+found directory errors

NO lost+found DIRECTORY
CREATE? There is no lost+found directory in the root directory of the file system; fsck tries to create one.

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.

`lost+found IS NOT A DIRECTORY`
`REALLOCATE?` The entry for `lost+found` is not a directory.

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 SPACE LEFT IN /lost+found`
`EXPAND?` There is no space to add another entry to the `lost+found` directory in the root directory of the file system.

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.

`DIR I=`*I1* `CONNECTED. PARENT WAS I=`*I2*
This advisory message indicates that a directory inode *I1* was successfully connected to the `lost+found` directory. The parent inode *I2* of the directory inode *I1* is replaced by the inode number of the `lost+found` directory.

`DIRECTORY` *f* `LENGTH` *s* `NOT MULTIPLE OF` *b*
`ADJUST?` A directory *f* has been found with size *s* that is not a multiple of the directory blocksize *b* (this can reoccur in Phase 3 if it is not adjusted in Phase 2).

Possible responses to the ADJUST? prompt are

Y The length is rounded up to the appropriate block size. This error can occur on UFS file systems. A warning is printed and the directory is adjusted.

N Ignores the error condition.

BAD INODE *S* TO DESCEND

An internal error has caused an impossible state *s* to be passed to the routine that descends the file-system directory structure. The `fsck` program exits. See a technical expert.

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 are correctable if the file system is being checked, except for running out of space in the `lost+found` directory.

Unreferenced files

UNREF FILE I=*i* OWNER=*o* MODE=*m* SIZE=*s* MTIME=*t*

RECONNECT? Inode *i* was not connected to a directory entry when the file system was traversed. The owner *o*, mode *m*, size *s*, and modify time *t* of inode *i* are printed. If the 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 by zeroing its contents.

N Ignores this error condition.

`lost+found` directory errors

`NO lost+found DIRECTORY`
`CREATE?` 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 `mklost+found`(1M) in *A/UX System Administrator's Reference.*

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.

N Stops the attempt to link up the lost inode. This action always invokes the `UNREF` error condition in Phase 4.

`lost+found IS NOT A DIRECTORY`
`REALLOCATE?` The entry for `lost+found` is not a directory.

Possible responses to the `REALLOCATE?` prompt are

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.

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?` 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`.

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 Stops the attempt to link up the lost inode. This action always invokes the `UNREF` error condition in Phase 4.

Incorrect free inode counts

The superblock contains a count of the total number of free inodes in the file system. The `fsck` program compares this count to the number of inodes 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 inode count.

`LINK COUNT FILE I=`*i* `OWNER=`*o* `MODE=`*m* `SIZE=`*s* `MTIME=`*t* `COUNT=`*x*
`SHOULD BE` *y*
`ADJUST?` 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`.

Possible responses to the `ADJUST?` prompt are

Y Replaces the link count of file inode *i* with *y*.

N Ignores this error condition.

Unreferenced files and directories

`UNREF FILE I=`*i* `OWNER=`*o* `MODE=`*m* `SIZE=`*s* `MTIME=`*t*

`CLEAR?` 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* by zeroing its contents.

`N` Ignores this error condition.

Bad and duplicate blocks in files and directories

`BAD/DUP FILE I=`*i* `OWNER=`*o* `MODE=`*m* `SIZE=`*s* `MTIME=`*t*

`CLEAR?` 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* by zeroing its contents.

`N` Ignores this error condition.

Phase 5: Check cylinder groups

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 the total used-inode count incorrect.

CG *c*: BAD MAGIC NUMBER

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.

BLK(S) MISSING IN BIT MAPS
SALVAGE? A cylinder group block map is missing some free blocks; the maps are reconstructed. Possible responses to the SALVAGE? prompt are

Y Reconstructs the free block map.

N Ignores this error condition.

SUMMARY INFORMATION BAD
SALVAGE? The summary information was found to be incorrect. Possible responses to the SALVAGE? prompt are

Y Reconstructs the summary information.

N Ignores this error condition.

FREE BLK COUNT(S) WRONG IN SUPERBLOCK
SALVAGE? The superblock free block information found to be incorrect is recomputed. Possible responses to the SALVAGE? prompt are

Y Reconstructs the superblock free block information.

N Ignores this error condition.

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)

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.

`***** REBOOT A/UX! **********`
This advisory message indicates that the root file system has been modified by `fsck`. A/UX reboots the system.

`***** 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 rebooted. A/UX reboots the system.

`fsck` initialization phase messages: SVFS-specific

Before a file-system check can be performed, certain tables have to be set up in memory and certain files have to be opened. This is the "initialization phase" of `fsck`. During this phase, `fsck` may discover error conditions that result from errors in command line options, memory requests, file opening, file status, file-system size checks, and scratch file creation.

`fsck` option errors

The following messages may appear when you specify the `fsck` command line incorrectly. See `fsck`(1M) in *A/UX System Administrator's Reference* for further details.

c`option?` *c* stands for any character that is not a legal flag option to `fsck`. Legal options are `y`, `n`, `s`, `S`, `q`, `D`, and `t`; `fsck` terminates on this error condition.

`Bad -t option`
The `t` option was not followed by a filename; `fsck` terminates on this error condition.

`Invalid -s argument, defaults assumed`
This is only a warning. The `s` option was not followed by `3`, `4`, or *blocks-per-cylinder*`:` *blocks-to-skip*. The `fsck` program assumes a default value of 400 blocks per cylinder and 9 blocks to skip.

`Incompatible options: -n and -s`
It's not possible to salvage the free list without modifying the file system; `fsck` terminates on this error condition.

Memory request errors

The following messages may appear when `fsck` detects errors in memory allocation requests. These messages may indicate a serious problem that you may not be able to solve without technical assistance.

`can't fstat standard input`
: The attempt to use `fstat` as standard input failed; `fsck` terminates on this error condition.

`can't get memory`
: The `fsck` program can't find the memory space it needs for its virtual memory tables; `fsck` terminates on this error condition.

Errors in opening files

The following messages may appear when `fsck` cannot open a file or file system:

`can't open:` *f*
: The default file-system check file *f* (`/etc/fstab`) cannot be opened for reading; `fsck` terminates on this error condition. Check the access modes of *f* and modify accordingly.

`can't open` *f*
: The file system *f* cannot be opened for reading. The `fsck` program ignores this file system and continues checking the next file system given. Check the access modes of *f* and modify accordingly.

f `is not a block or character device`
: The `fsck` program has been given a regular filename by mistake; `fsck` ignores this file system and continues checking the next file system given. Check the file type of *f* and modify accordingly.

File status errors

The following messages may appear when `fsck` cannot obtain a file's status. These errors may indicate a serious problem that you may not be able to solve without technical assistance.

`can't stat root`
: The `fsck` program request for statistics about the root directory (`/`) failed; `fsck` terminates on this error condition.

`can't stat` *f*
: The `fsck` program request for statistics about the file system *f* failed. It ignores this file system and continues checking the next file system given. Check the access modes of *f*.

File-system size and inode list size

The file-system size and inode list size are critical pieces of information to the `fsck` program. Although `fsck` can't actually check these sizes, it can check if they're within reasonable bounds. The file-system size must be larger than the number of blocks used by the superblock and the number of blocks used by the list of inodes. The number of inodes must be less than 65,535. All other file-system checks depend on the correctness of these sizes. (Note that UFS files have a larger number of inodes than do SVFS files.)

You may see the following message when `fsck` detects an inconsistency in the file-system size:

`Size check: fsize` *x* `isize` *y*
: 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.

Scratch file errors

A **scratch file** is a temporary file that `fsck` creates when you invoke `fsck` with the `t` flag option. This option may be necessary if there is not enough core memory to hold `fsck`'s tables. You may see this message if `fsck` cannot create its own scratch file for a particular file system:

`can't create` *f*
: This message means that `fsck`'s request to create a scratch file *f* failed. It ignores this file system and continues checking the next file system given. Check access modes of *f*.

Interactive messages

These messages require your yes or no response to the `CONTINUE?` prompt. "Yes" may be `Y` or `y`, and "no" may be `N` or `n`; these responses are shown in uppercase here. The following messages may indicate a serious problem because the file system cannot be completely checked. You may need to obtain technical assistance.

`CANNOT SEEK: BLK` *b*
`CONTINUE?`
: 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. If the block *b* was part of the virtual memory buffer cache, `fsck` will terminate with the message: `Fatal I/O error.`

N
: Stops `fsck`.

`CANNOT READ: BLK` *b*
`CONTINUE?`
: The `fsck` program request for reading 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. Try a second run of `fsck`. If the block *b* was part of the virtual memory buffer cache, `fsck` terminates with the message `Fatal I/O error.`

N
: Stops `fsck`.

`CANNOT WRITE: BLK` *b*
`CONTINUE?` The `fsck` program request for writing a specified block number *b* in the file system failed. The disk is write-protected.

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. If the block *b* was part of the virtual memory buffer cache, `fsck` terminates with the message `Fatal I/O error`.

N Stops `fsck`.

Phase 1: Check blocks and sizes

During this phase of execution, `fsck` 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. The list of inodes is checked sequentially, starting with inode 1 (there is no inode 0) and going to the last inode in the file system.

Although each individual inode has only a small chance of becoming inconsistent, there are so many of them that it's almost as likely that an inconsistency will occur in the inode list as in the superblock.

Inode type errors

Each inode contains a mode word that describes the type and state of the inode.

Inodes may be one of five types: regular, directory, special block, special character, or FIFO, according to the file involved. If an inode is not one of these types, it is of an illegal type. If this is the case, tell `fsck` to clear the inode.

`UNKNOWN FILE TYPE I=`*i*

`CLEAR?` 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 strange occurrence with the mode bits.

Possible responses to the `CLEAR?` prompt are

Y Deallocates inode *i* by zeroing its contents. This always invokes the `UNALLOCATED` error condition in Phase 2 for each directory entry pointing to this inode.

N Ignores this error condition.

Zero-link-count table errors

Each inode contains a stored count of the total number of directory entries linked to the inode. The `fsck` program verifies the link count of each inode by traversing down the total directory structure, starting from the root directory, and calculating an actual link count for each inode.

If the stored link count is nonzero and the actual link count is zero, no directory entry appears for the inode. If the stored and actual link counts are nonzero and unequal, a directory entry may have been added or removed without the inode being updated.

If the stored link count is nonzero and the actual link count is zero, `fsck` can link the disconnected file to the `lost+found` directory (at your direction). If the stored and actual link counts are nonzero and unequal, `fsck` can replace the stored link count with the actual link count.

`LINK COUNT TABLE OVERFLOW`

`CONTINUE?` An internal table for `fsck` has no more room. This is a rare error. Contact your Apple representative for assistance.

Possible responses to the `CONTINUE?` prompt are

Y Continues with the program. This error condition makes a complete check of the file system impossible. You should make a second run of `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`.

Bad or duplicate blocks

Each inode contains a list of pointers to lists (indirect blocks) of all the blocks claimed by the inode.

The `fsck` program checks each block number claimed by an inode for a value lower than that of the first data block or greater than that of the last block in the file system. If the block number is outside this range, it is a bad block number. If an indirect block was not written to disk, an inode may contain many bad blocks. In this case, `fsck` clears both inodes (at your direction).

The `fsck` program compares each block number claimed by an inode to a list of already allocated blocks. If a block number is already claimed by another inode, the block number is added to a list of duplicate blocks. Otherwise, the list of allocated blocks is updated to include the block number. If there are any duplicate blocks, `fsck` makes a partial second pass of the inode list to find the inode of the duplicated block. This is necessary because without examining the files associated with these inodes for correct content, `fsck` does not have enough information to decide which inode is corrupted and should be cleared. Usually, the inode with the earliest modify time is incorrect and should be cleared. This condition may be the result of a file system containing blocks claimed by both the free list and other parts of the file system.

A large number of duplicate blocks in an inode may be due to an indirect block not being written to disk. In this case, `fsck` clears both inodes (at your direction).

b `BAD I=`*i*

The variable *i* stands for the actual number on your screen. Inode *i* contains block number *b*, which is 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 if inode *i* has too many block numbers outside the file-system range.

This error condition always invokes the `BAD/DUP` error condition in Phase 2 and Phase 4.

`EXCESSIVE BAD BLKS I=`*i*

`CONTINUE?` As before, *i* stands for the actual number on your screen. 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. Usually, ten is the 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 makes a complete check of the file system impossible. A second run of `fsck` should be made to recheck this file system.

N Stops `fsck`.

b `DUP I=`*i* 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.

`EXCESSIVE DUP BLKS I=`*i*

`CONTINUE?` As before, *i* stands for the actual number on your screen. The number of blocks claimed by other inodes is too high to be acceptable. Usually, ten is the 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. A second run of `fsck` should be made to recheck this file system.

N Stops `fsck`.

`DUP TABLE OVERFLOW`

`CONTINUE?` An internal table in `fsck`, containing duplicate block numbers, has no more room.

Possible responses to the `CONTINUE?` prompt are

Y Continues with the file-system check. This error condition makes a complete check of the file system impossible. Try a second run of `fsck`. If another duplicate block is found, this error condition is repeated.

N Stops `fsck`.

Inode size errors

Each inode contains a 32-bit (4-byte) field. This size indicates the number of characters in the file associated with the inode. This size can be checked for inconsistencies; for example, directory sizes are not a multiple of 16 characters, or the number of blocks actually used does not match the number indicated by the inode size.

A directory inode within the file system has the directory bit on in the inode mode word. The directory size must be a multiple of 16 because a directory entry contains 16 bytes (2 bytes for the inode number and 14 bytes for the file or directory name). The `fsck` program warns of such directory misalignment. This is only a warning, because not enough information can be gathered to correct the misalignment.

A rough check of an inode's size field consistency can be performed by computing from the size field the number of blocks that should be associated with the inode and comparing that number to the actual number of blocks claimed by the inode. The `fsck` program calculates the number of blocks by dividing the number of characters in an inode by the number of characters per block and rounding up. The `fsck` program adds one block for each indirect block associated with the inode. If the actual number of blocks does not match the computed number of blocks, `fsck` warns of a possible file-size error. This is only a warning, because the system does not fill in blocks in files created in random order.

- *Note:* These messages are only warnings. If you use the `q` option on the `fsck` command line, these messages are not printed.

`POSSIBLE FILE SIZE ERROR I=`*i*

Again, *i* stands for the actual number on your screen. The inode size does not match the actual number of blocks used by the inode. 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.

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 it into 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` would not do the job.

`DIRECTORY MISALIGNED I=`*i*

Again, *i* stands for the actual number on your screen. The size of a directory inode is not a multiple of the size of a directory entry (usually 16). The directory inode should be cleared.

Inode format errors

Each inode contains a mode word that describes the type and state of the inode. Inodes may be found in one of three states: allocated, unallocated, or neither allocated nor unallocated. This last state indicates an incorrectly formatted inode. An inode can get in this state if bad data is written into the inode list, a possible result of a hardware failure. The only corrective action is for `fsck` to deallocate the inode.

`PARTIALLY ALLOCATED INODE I=`*i*

`CLEAR?` Inode *i* is neither allocated nor unallocated.

Possible responses to the `CLEAR?` prompt are

Y Deallocates inode *i* by zeroing its contents.

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. If the duplicate block is found, the following error condition occurs:

b `DUP I=`*i*
: Inode *i* contains block number *b*, which is already claimed by another inode. This error condition always invokes the `DUPS/BAD` error condition in Phase 2. You can determine which nodes have overlapping blocks by examining this error condition and the `DUP` error condition in Phase 1.

Phase 2: Check pathnames

This phase removes directory entries pointing to inodes that had error conditions from Phase 1 and Phase 1B. In Phase 2, `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.

Root inode mode and status errors

`ROOT INODE UNALLOCATED. TERMINATING`
: 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 technical assistance. The program ends.

`ROOT INODE NOT DIRECTORY`
`FIX?`
: The root inode (usually inode number 2) is not of the directory type.

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`.

`DUPS/BAD IN ROOT INODE`
`CONTINUE?` 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.

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.

N Stops `fsck`.

Directory inode pointers range errors

`I OUT OF RANGE I=`*i* `NAME=`*f*
`REMOVE?` A directory entry *f* has an inode *i* greater than the end of the inode list.

Possible responses to the `REMOVE?` prompt are

Y Removes the directory entry *f*.

N Ignores this error condition.

Directory entries pointing to bad inodes

`UNALLOCATED I=`*i* `OWNER=`*o* `MODE=`*m* `SIZE=`*s* `MTIME=`*t* `NAME=`*f*
`REMOVE?` 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.

Possible responses to the `REMOVE?` prompt are

Y Removes the directory entry *f*.

N Ignores this error condition.

`DUPS/BAD I=`*i* `OWNER=`*o* `MODE=`*m* `SIZE=`*s* `MTIME=`*t* `DIR=`*f*
`REMOVE?` During Phase 1 or Phase 1B, `fsck` has found duplicate blocks or bad blocks associated with directory entry *f*, directory inode *i*. The owner *o*, mode *m*, size *s*, modify time *t*, and directory name *f* are printed.

Possible responses to the `REMOVE?` prompt are

Y Removes the directory entry *f*.

N Ignores this error condition.

`DUPS/BAD I=`*i* `OWNER=`*o* `MODE=`*m* `SIZE=`*s* `MTIME=`*t* `FILE=`*f*
`REMOVE?` 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*.

N Ignores this error condition.

`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*.

The `fsck` program checks directory blocks for nonzero padded entries, inconsistent `.` and `..` entries, and embedded slashes in the name field. 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.

Phase 3: Check connectivity

During Phase 3, `fsck` 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.

Unreferenced directories

`UNREF DIR I=`*i* `OWNER=`*o* `MODE=`*m* `SIZE=`*s* `MTIME=`*t*
`RECONNECT?` The directory inode *i* was not connected to a directory entry when the file system was traversed. The owner *o*, mode *m*, size *s*, and modify time *t* of the directory inode *i* are printed. The `fsck` program forces the reconnection of a nonempty directory.

Possible responses to the `RECONNECT?` prompt are

Y Reconnect the directory inode *i* to the file system in the directory for lost files, usually `lost+found`. This may invoke `lost+found` error conditions (described later) if there are problems connecting the directory inode *i* to `lost+found`. This may also invoke the `CONNECTED` error condition (described later) if the link was successful.

N Ignores this error condition. This action always invokes the `UNREF` error condition (described later).

`lost+found` directory errors

`SORRY. NO lost+found DIRECTORY`
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 access modes. See `fsck`(1M) and `mklost+found`(1M) in *A/UX System Administrator's Reference* for further details.

`SORRY. NO SPACE IN lost+found DIRECTORY`
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.

`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.

Phase 4: Check reference counts

During Phase 4, `fsck` checks the directory connectivity seen in Phase 2 and Phase 3. In Phase 4, `fsck` reports errors 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.

Unreferenced files

`UNREF FILE I=`*i* `OWNER=`*o* `MODE=`*m* `SIZE=`*s* `MTIME=`*t*

`RECONNECT?` Inode *i* was not connected to a directory entry when the file system was traversed. The owner *o*, mode *m*, size *s*, and modify time *t* of inode *i* are printed. If the 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 later) if there are problems connecting inode *i* to `lost+found`.

N Ignores this error condition. This action always invokes the `CLEAR?` error condition (described later).

`lost+found` directory errors

`SORRY. NO lost+found DIRECTORY`
: There is no `lost+found` directory in the root directory of the file system; `fsck` ignores the request to link a file in `lost+found`. This action always invokes the `CLEAR?` error condition (described later). Check the access modes of `lost+found`. Also see `mklost+found`(1M) in *A/UX System Administrator's Reference.*

`SORRY. NO SPACE IN lost+found DIRECTORY`
: There is no space to add another entry to the `lost+found` directory in the root directory of the file system; `fsck` ignores the request to link the file. This action always invokes the `CLEAR?` error condition (described next). Check the size and contents of `lost+found`.

`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 by zeroing its contents.

N
: Ignores this error condition.

Incorrect free inode counts

The superblock contains a count of the total number of free inodes in the file system. The `fsck` program compares this count to the number of inodes 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 inode count.

`LINK COUNT FILE I=`*i* `OWNER=`*o* `MODE=`*m* `SIZE=`*s* `MTIME=`*t* `COUNT=`*x*
`SHOULD BE` *y*
`ADJUST?`
: 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.

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?` 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.

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?` 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.

Possible responses to the `ADJUST?` prompt are

Y Replace the link count of inode *i* with *y*.

N Ignores this error condition.

Unreferenced files and directories

`UNREF FILE I=`*i* `OWNER=`*o* `MODE=`*m* `SIZE=`*s* `MTIME=`*t*
`CLEAR?` 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 the `CLEAR?` prompt are

Y Deallocates inode *i* by zeroing its contents.

N Ignores this error condition.

`UNREF DIR I=`*i* `OWNER=`*o* `MODE=`*m* `SIZE=`*s* `MTIME=`*t*

`CLEAR?` 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.

Possible responses to the `CLEAR?` prompt are

Y Deallocates inode *i* by zeroing its contents.

N Ignores this error condition.

Bad and duplicate blocks in files and directories

`BAD/DUP FILE I=`*i* `OWNER=`*o* `MODE=`*m* `SIZE=`*s* `MTIME=`*t*

`CLEAR?` 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* by zeroing its contents.

N Ignores this error condition.

`BAD/DUP DIR I=`*i* `OWNER=`*o* `MODE=`*m* `SIZE=`*s* `MTIME=`*t*

`CLEAR?` During Phase 1 or Phase 1B, `fsck` has 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.

Possible responses to the `CLEAR?` prompt are

Y Deallocates inode *i* by zeroing its contents.

N Ignores this error condition.

`FREE INODE COUNT WRONG IN SUPERBLK`
`FIX?` 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.

Possible responses to the `FIX?` prompt are

`Y` Replaces the count in superblock with the actual count.

`N` Ignores this error condition.

Phase 5: Check free list

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.

`EXCESSIVE BAD BLKS IN FREE LIST`
`CONTINUE?` 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 in Phase 5.

N Stops `fsck`.

`EXCESSIVE DUP BLKS IN FREE LIST`
`CONTINUE?` 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 in Phase 5.

N Stops `fsck`.

`BAD FREEBLK COUNT`
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 in Phase 5.

x `BAD BLKS IN FREE LIST`
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 in Phase 5.

x `DUP BLKS IN FREE LIST`
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 in Phase 5.

x `BLK(S) MISSING`
The free list does not contain *x* blocks unused by the file system. This error condition always invokes the `BAD FREE LIST` condition in Phase 5.

`FREE BLK COUNT WRONG IN SUPERBLK`
`FIX?` 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.

`BAD FREE LIST`

`SALVAGE?` 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.

Phase 6: Salvage free list

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 modified status.

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.

`***** 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 reboot because it can fix the file system image only on the disk but not in memory. Since A/UX periodically issues a `sync`(1) call to update the file system from memory, the forced reboot of the system prevents the file-system repair from being undone by the automatic `sync` call.

`***** FILE SYSTEM WAS MODIFIED **********`

This advisory message indicates that `fsck` has modified a nonroot file system to fix the inconsistencies it found. A/UX continues to run.

Chapter 9 **System Accounting Package**

Two packages provided with your A/UX system allow you to keep track of the details of system operation and usage:

- The **system accounting package** collects information and generates reports on buffer activity, CPU utilization in general, device activity, and so on. This chapter discusses the system accounting package.
- The **system activity package** permits you to keep track of all low-level activity in your system. For more information, see Chapter 10, "System Activity Package."

The system accounting package collects detailed information on system usage and allows you to generate a comprehensive system usage report on a daily and monthly basis. (Note that the report provides information on overall system usage, not on individual users.) This report function, for the most part, is automatic and is described in the next section, "Routine Accounting Procedures." You can also produce customized reports by issuing accounting commands; these are described in "Special Accounting Procedures: `acctcom`," later in this chapter.

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. Take these steps to turn on system accounting and automate its operation:

1. **Log in as the root user.**

2. **Make sure the following lines are in the `/etc/rc` file and are not commented out (if they are preceded by a number sign [#], remove it):**

   ```
   /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. **Make sure that the following lines in the `/usr/spool/cron/crontabs/adm` file are not commented out (if they are preceded by a number sign [#], remove it):**

   ```
   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
   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 -uybd
   ```

   - *Note:* The first two lines and the last two lines in the example *must* appear as one line in the `adm` file. These lines are broken here only because the book page does not accommodate the long measure.

The adm file instructs cron to run the daily accounting automatically. The line

```
15 5 1 * * /usr/lib/acct/monacct
```

uses the monacct file to clean up all daily reports and daily total accounting files. The adm file also deposits one monthly total report and one monthly total accounting file in the fiscal directory after runacct has completed the last day's entries. By default, monacct uses the current month's date as the suffix for the filenames.

4. **Make sure 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.

The cron program

You don't need to understand how cron works in order to use the accounting package. Understanding the crontab file format, however, makes it easier for you to modify crontabs used for this package. This section provides a quick introduction to cron.

The cron program performs specific tasks at specified times. This information is stored in the user's crontab file /usr/spool/cron/crontabs/*login-name* and/usr/spool/cron/crontabs/adm, which is read and acted on by cron. Each line of text in this file is a small script consisting 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,

1. *minute* Time in minutes (0–59)
2. *hour* Time in hours (0–23)
3. *day-of-month* Day of the month (1–31)
4. *month-of-year* Month of the year (1–12)
5. *day-of-week* Day of the week (0–6, with 0 as Sunday)

Each of these fields may contain either an asterisk, which means ignore all cases, or a list of elements separated by commas, which specify specific cases to be activated. 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).

In this example, the line

```
15 5 1 * * /usr/lib/acct/monacct
```

means this: At the 15th minute of the 5th hour on the 1st day of each month, run the `monacct` program, which is located in the `/usr/lib/acct` directory. The asterisks in the *month-of-year* and *day-of-week* fields tell the system to ignore these fields.

You can change the field entries in the file `/usr/spool/cron/crontabs/adm` and run the `crontab` command on it, for instance, to run the accounting procedures more frequently or even to print a weekly report instead of a monthly one.

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 fields of four digits each:

  yyyy hhmm hhmm (number of spaces irrelevant)

  - 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.

The lines of text you enter in the /usr/spool/cron/crontabs file cause the system to automatically run the startup, ckpacct, turnacct, dodisk, runacct, and monacct procedures. Each of these six procedures is described in this section. Only some of the procedures are available. Other procedures—including acctcom—are also described. If you add these to the /usr/spool/cron/crontabs/adm file and run the crontab command on the edited file, they are processed automatically. For example, acctcom is discussed in "Special Accounting Procedures: acctcom," later in this chapter.

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.
- Turnacct 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 saved pacct and wtmp files that runacct creates.

The ckpacct procedure

After process accounting has begun, cron 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 1000 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 turnaccct 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.

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] [*file1...*]

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 *file1* specifies the name or names of the file system or file systems in which the disk accounting is done. If you use the *files* argument, disk accounting is performed on these file systems only. If you use the `-o` flag, *files* should be the names of the directories on which the file systems are mounted. If you omit the `-o` flag, *files* should be the special filenames of mountable file systems.

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 ($2.00) 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 `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 [`*mmdd*`] [`*mmdd* `state]`

You can use the options to restart `runacct` after a failure. They are explained later in this section.

The entries made to the `/usr/spool/cron/crontabs/adm` file initiate 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.

To read the daily report, type

```
/usr/lib/acct/prdaily | more
```

The `prdaily` reporting function is covered in more detail later in this chapter. The preceding command prints 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,

```
Oct 26 10:04 1987 DAILY REPORT FOR A/UX Page 1

from Tue Oct 20 04:00:13 1990
to Tue Oct 20 04:00:13 1990
```

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 acctcon1
```

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. The fields in this part of the report are

`TOTAL DURATION` The amount of time the system was in multi-user mode.

`LINELINE` 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 the multiplexer, modem, or cable is

going bad. You should monitor the `/etc/wtmp` file. If it grows rapidly, execute `acctcon1` to see which `tty` 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," earlier 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 `acctdusg` program. The `# OF PROCS` column gives the number of processes used. A very high number might indicate a shell process gone wild. The `# OF SESS` column tells you how many times each user logged in.

`DISK SAMPLES` The number of times the `acctdusg` ran to obtain the information in the `DISK BLOCKS` column.

`FEE` A record of disk charges to each user ID. This information is written to `/usr/adm/fee` 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. You can fine-tune the system by making commonly used commands more accessible.

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.

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.

  - *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 superuser privileges. See "Special Accounting Procedures: acctcom," 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 `TOTAL KCOREMIN` over `NUMBER CMNDS`.

`MEAN CPU-MIN`
: Calculated as `NUMBER CMNDS` over the `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:

```
Sep 29 04:05 1990 LAST LOGIN Page 1

90-09-25 apple
90-09-27 alice
00-00-00 phil
00-00-00 sys
90-09-29 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.

The `prdaily` procedure

The script `prdaily` generates a printout of the `runacct` process. The report resides in `/usr/adm/acct/sum/rprt`*mmdd*. The command syntax is

`/usr/lib/acct/prdaily [-l] [-c] [`*mmdd*`]`

The notation *mmdd* 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`. Remember, the daily information is no longer available after `monacct` is run. The `-l` flag prints a report of exceptional usage by login identification for a date specified with *mmdd*. The `-c` flag prints a report of exceptional resource usage by command. You can use it on the current day's accounting data only. These values are considered to signal exceptional usage: `CPU` > 80, `KCORE` > 500, and `CONNECT` > 120.

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 the file `/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` *mmdd* 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`.**
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 `wtmp` file and corrects date changes if necessary.
`CONNECT1`	Produces connect session records in `ctmp.h` format.
`CONNECT2`	Converts `ctmp.h` format files into `tacct.h` format.
`PROCESS`	Converts process accounting records into `tacct.h` format.
`MERGE`	Merges the connect and process accounting records.
`FEES`	Converts the output of `chargefee` into `tacct.h` 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 daytacct with the summary total accounting records in /usr/adm/acct/sum/tacct.
CMS	Produces the command summaries.
USEREXIT	You can include customized accounting procedures here.
CLEANUP	Cleans up the temporary files and exits. COMPLETE appears in this file when runacct 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 *mmdd*, 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 acctcms -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: locks found, run aborted`
: The files `lock` and `lock1` were found. You must remove them to restart `runacct`.

`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: turnacct switch returned rc= ?`
: Check the integrity of `turnacct` and `accton`. The `accton` program must be owned by `root` and have the set-uid bit set.

`ERROR: Spacct?.`*mmdd*`already exists`
: File setups probably have already been run. Check the status of files and run setups manually.

`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.

`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.

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:

```
cd /usr/adm/acct/nite
fwtmp < wtmp.mmdd > xwtmp
ed xwtmp
```

and delete the corrupted records, or delete all records from beginning up to the date of change:

```
fwtmp -ic < xwtmp > wtmp.mmdd
```

If you can't fix the `wtmp` file, create a null `wtmp` file, which will prevent connect time from being charged incorrectly. The `actprc1` 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:

```
cd /usr/adm/acct/sum
acctmerg -v < tacct.mmdd > xtacct
ed xtacct
```

Next, remove the bad records and write duplicate UID records to another file:

```
acctmerg -i < xtacct > tacct.mmdd
acctmerg tacctprev < tacct.mmdd > tacct
```

- *Note:* You can recreate `sum/tacct` by merging all the `tacct.`*mmdd* files (the `monacct` procedure does this).

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 "The `cron` Program" 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`.

Special accounting procedures: `acctcom`

Besides the user information you can get from the automated procedures, additional information about users is available. For example, the automated procedure 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 `acctcom` command is 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 `acctcom` is

`acctcom [options] [file]`

The `acctcom` 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 superuser privileges. By using the `-n` option of the command, you can find out which users (selected with the `-u` 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 `read` and `write` 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 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 `acctcom` 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 `fork/exec` 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 `HOG FACTOR`.

`-l` This option is not extremely useful, given the state of `tty` information. 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 *time*.
`-S` selects processes starting at or after *time*.
`-e` selects processes existing at or before *time*.
`-E` selects processes ending at or before *time*.

Chapter 10 **System Activity Package**

Two packages provided with your A/UX system allow you to keep track of the details of system operation and usage:

- The **system accounting package** collects information and generates reports on buffer activity, CPU use in general, device activity, and so on. For more information, see Chapter 9, "System Accounting Package."
- The **system activity package** permits you to keep track of all low-level activity in your system. This chapter discusses the system activity package.

The system activity package 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 to 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.

The system activity counters

A series of system activity counters must be working to determine what processes are being run 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. 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 Chapter 9, "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. Particular emphasis is placed on those counters that you can use to troubleshoot or fine-tune your system.

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 is used to collect 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 can be used 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 rebooted. When the arguments *t* and *n* are used, sadc samples the counters *n* times every *t* seconds and writes the data in binary format to the named *file* or, by default, to the standard output, /usr/adm/sa/sa*dd*, where *dd* stands for a given day.

The sa1 and sa2 commands

The sa1 and sa2 commands supply the default parameters for the operation of sadc and sar.

The sa1 command invokes sadc to enter the information gathered by the system counters at the intervals specified in the /usr/lib/cron/crontab file into the daily data file /usr/adm/sa/sa*dd*. This information is in binary format.

The sa2 command is a variation of 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 sar*dd*. Again, *dd* stands for the day of the report. You can use the sar options with the sa2 command. For a complete explanation of these options, see "The sar Command Options," later in this chapter.

Setting up the system activity functions

It is a good idea to monitor and record system activity routinely in a standard way for historical analysis. Each of the previous commands initiates activity observations. For these observations to occur, you must first initiate the data collection functions. By automating these functions, you can generate regular system activity reports.

Your system can automatically sample the activity counters and store the information in a file for later reporting.

To do this, edit the following lines in `/usr/spool/cron/crontabs/adm` so that they are "commented out." This is done by placing a number sign before each line:

```
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
```

Next, run the `crontab` command on this newly modified file. This produces records every 20 minutes during working hours and hourly otherwise.

You can generate hourly records during working hours by substituting

```
5 18 * * 1-5 /usr/lib/sa/sa2 -s 8:00 -e 18:01 -i3600 -A
```

for

```
0 18-7 * * 1-5 /usr/lib/sa/sa1
```

These entries cause the collection functions to operate. They do not, however, generate reports automatically. It is still the responsibility of the system administrator to check the desired reporting function regularly and to generate the report for current and later review.

The system activity report commands

The three commands `sar`, `sag`, and `timex` generate system activity reports.

You can use these commands to observe system activity during

- normal operations
- a controlled stand-alone test of a large system
- 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 `sag` command displays system activity in graph form.

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.

These commands and their options and applications are discussed in detail in the following sections.

The `sar` command

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. The output should look like this:

```
00:00:03    %usr        %sys        %wio        %idle
01:00:03    1           1           0           98
02:00:03    1           1           0           98
14:39:12    41          12          4           42
14:59:12    3           5           3           89
15:20:04    14          12          4           70
15:40:04    11          8           3           78
Average     5           5           1           89
```

The column headings display the following:

`%usr`	amount of time running in user mode
`%sys`	amount of time running in system mode
`%wio`	idle time with process waiting for block I/O
`%idle`	idle time not waiting for block I/O

If you enter

```
sar > sar.output
```

the output from `sar` is stored in the file `sar.output`. It doesn't scroll across your screen.

By entering `sar` with the options discussed in the next section, 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 [-u] [-b] [-d] [-w] [-c] [-a] [-q] [-v] [-m] [-A] [-o`*file*`] t [`*n*`]`

`sar [-u] [-b] [-d] [-w] [-c] [-a] [-q] [-v] [-m] [-A] [-s`*time*`] [-e`*time*`] [-i`*sec*`] [-f`*file*`]`

The first word is the command itself; the bracketed letters within the brackets are flag options that sample different counters. The remaining options allow you to sample certain counters at specified times and save the results for later viewing.

In the command syntax

`sar [-u] [-b] [-d] [-w] [-c] [-a] [-q] [-v] [-m] [-A] [-s`*time*`] [-e`*time*`] [-i`*sec*`] [-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/sa`*dd,* where *dd* stands for a given 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, 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 activity at intervals of 3600 seconds, which means it samples activity at intervals of 3600 seconds, or every hour starting at 8:00 A.M. and ending at 6:01 P.M.

In the command syntax

`sar [-u] [-b] [-d] [-w] [-c] [-a] [-q] [-v] [-m] [-A] [-o`*file*`] t [`*n*`]`

`sar` invokes the data collection program `sadc` to sample the system activity counters every *t* seconds for *n* intervals and generates a system activity report.

If you specify the `-o` option, `sar` saves the output in binary format into *file.*

If you supply no frequency arguments, `sar` generates system activity reports for the time interval specified in the existing data file. On this system, that time period is from 8:00 to 18:00, and the data is sampled every hour. See "Setting Up the System Activity Functions," earlier in this chapter. Unless you redirect the output to a file, it scrolls across the screen.

- *Note:* All reports you generate with the options listed 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
```

The system activity package automatically generates a report containing all of the options listed in the following sections. The report is stored in the `/usr/adm/sa` directory, in the files listed as `sar`*dd.* These files are the binary representations of the `sar` reports. You can get the same report by typing

```
sar -A
```

This generates the same information as

```
sar -udqbwcayvm
```

but takes fewer keystrokes.

The `sar` command options

The options you can use with the `sar` command are listed in the following subsections.

The `-u` option

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 display the following:

`%usr`	amount of time running in user mode
`%sys`	amount of time running in system mode
`%wio`	idle time with process waiting for block I/O
`%idle`	idle time 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 (a count on the system), which occurs 60 times per second.

The `-b` option

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, brwrit/s`
: Samples the `bread` and `brwrite` 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 if files are being accessed from the buffers or if the information has to be retrieved from the disk itself. A low ratio might suggest that more efficient buffering could increase system response time in a certain area.

The -d option

The `-d` option reports 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 or tape 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 or tape drive.
`blks/s`	Displays the number of bytes transferred and counted in block-sized units.
`avwait`	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 `avwait` 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. By time ticks, they measure the time it takes a device to receive, process, and transmit a request, summed over all I/O requests for the device. 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 and tape devices are being put to their best use. For example, if one device 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 device. Also, if the active time counter for a device 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.

The -w option

The `-w` option reports swapping and switching activity. This option uses the `swapin` and `swapout` 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` are usually higher than those for `swpot`. This peculiar asymmetry arises from programs with the "sticky bit" set, which keeps a program on a contiguous area of the swap device. Therefore, moving the program back and forth between memory and the swap device is more efficient if the sticky bit is not set.

If you enter `sar -w`, the system generates a report organized in columns as follows:

`swpin/s` Reports the number of transfers from the swap area on disk to main memory.

`swpot/s` Reports the number of transfers from memory to the swap area.

`bswin/s, bswot/s`
Reports the number of bytes transferred.

`pswch/spswch/s`
Reports the number of process switches that have occurred.

The -c option

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.

The -a option

The -a option reports on use of file access system routines. Avoid using this option, because this information is usually inaccurate. 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:

`iget/s` Measures the number of requests for the inode number that corresponds to a particular file. The `iget` routine is used to locate the inode entry (i-number) of a file. See Chapter 8, "Checking the A/UX File System: `fsck`," for an explanation of i-numbers and inodes. The `iget` routine first searches the in-core (main memory) inode table. If the inode entry is not in the table, `iget` gets the inode from the file system where the file resides and makes an entry in the inode table.

`namei/s` Measures the number of requests for a file-system path search. The `namei` 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, `namei` calls `iget` to find the i-number of the file it is searching for. Therefore, counter `iget` is always greater than counter `namei`.

`dirblk/s` Measures and records the number of directory block read requests issued by the system. Dividing the directory blocks read by the number of `namei` 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.

The -q option

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.

The -v option

The -v option reports the status of text, process, inode, and file tables. 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.

The -m option

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.

The sag command

The system activity graph command is sag, which displays the system activity data that was 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.*

The timex command

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 children.

`-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.

Chapter 11 **Troubleshooting**

This chapter lists some common problems that A/UX users may experience, along with actions that you can take to identify and correct them.

- **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 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.
- **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 7).
  - System software may not be installed. Install, or reinstall, if necessary.
  - Boot blocks may be damaged. Reinstall system software.
  - 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.
- **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. Boot 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, or try removing `inits`, which are the contents of your initialization script.

- If the system fails to boot from the floppy disk, 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 on 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 ID, 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 on 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/c`*x*`d0s0`, 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 window 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 booting 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 Yellow Pages, make sure that the server is running.

- **You add an `init` or a `CDEV` to the System Folder that does not show up while booting A/UX, or else it does not work.**
  - Perhaps the `init` or `CDEV` files were not designed to show their icons during the startup process, or they weren't turned on in the Control Panel.
  - A/UX uses a different System Folder than does the Macintosh OS for `inits` and `CDEVs`. Place the appropriate files in `/mac/sys/SystemFolder` and log out. Then log in. The `CDEV` or `init` should be installed.
- **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, remove old files from the full file system. Make backups of files to be removed, either on tape or on floppy disks.
- **While you try to partition a disk for A/UX with Apple HD 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 vendor to get the right software.
  - A non-SCSI device, such as the Apple Hard Disk 20, cannot be read by Apple HD 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 NPROCs in increments of 25 until the message no longer appears. You must reboot each time you enter 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 the superuser and run the command again.

- **While trying to unmount a mounted file system, you encounter the error message : */filesystem*: `Device busy`.**
  - The file system 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 device 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/tc`*x*, 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 HD 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 `lpq` to verify that the printer is accepting requests. With the `lp` spooler, enter the `lpstat` command. Restart the scheduler `/usr/lib/lpsched`.

Index

B

C

D

H

I, J

K

L

M

N, O

P

Q

R

S

T

U

V

W

X

Y

Z

THE APPLE PUBLISHING SYSTEM

This Apple manual was written, edited, and composed on a desktop publishing system using Apple Macintosh® computers and Microsoft Word software. Proof pages were created on Apple LaserWriter® printers. Final pages were created on the Varityper VT600W imagesetter. Line art was created using Adobe Illustrator. POSTSCRIPT®, the page-description language for the LaserWriter, was developed by Adobe Systems Incorporated.

Text type and display type are Apple's corporate font, a condensed version of ITC Garamond. Bullets are ITC Zapf Dingbats®. Some elements, such as program listings, are set in Apple Courier.

Writers: A/UX Staff Writers

Developmental Editor: Jessie Wood

Production Supervisors: Josephine Manuele and Rex Wolf

Formatter: Roy Zitting

Special thanks to Vicki Brown, Li Greiner, David Payne, Eryk Vershen, and Chris Wozniak

030-0762-A