Apple®

A/UX™ Release Notes

Version 1.0

APPLE'S SYSTEM V IMPLEMENTATION A/UX RELEASE 1.0 RUNNING ON A MACINTOSH II COMPUTER HAS BEEN TESTED BY THE AT&T-IS' SYSTEM V VERIFICATION SUITE AND CONFORMS TO ISSUE 2 OF AT&T-IS' SYSTEM V INTERFACE DEFINITION BASE PLUS KERNEL EXTENSIONS.

Introduction to A/UX Release Notes, Version 1.0

These release notes contain late-breaking information about release 1.0 of the A/UX™ software for the Apple® Macintosh® II computer. This package contains two kinds of materials:

- Specific information that was not available in time to be incorporated into the printed manuals. Insert the pages that contain this information in the "Notes" section at the back of *A/UX Local System Administration.* Some of this material represents an expansion or update to a specific manual. In these cases, you may want to mark the table of contents for the affected manual with a reference to these release notes.
- Replacement command reference pages. Insert these pages alphabetically by section into the appropriate A/UX reference manuals (*A/UX Command Reference, A/UX Programmer's Reference,* and *A/UX System Administrator's Reference*).

You should read these release notes before installing and using your A/UX system, so that you can avoid potential problems caused by using out-of-date documentation.

Additional update information appears in the software itself, in the standard text file `/README`. You can read this file in either of two ways:

- Page through the text by entering the command

  ```
  more /README
  ```

  To see more information, press the space bar at the `--More--` prompt. When you reach the end of the text file, you are returned to the shell. To quit before the end of the file, press the Q key.
- Use your favorite A/UX text editor to read the text.

These release notes are divided into eight sections:

Section 1, "Programming Notes," contains a collection of implementation notes directed primarily at programmers.

Section 2, "System Administration," supplements information in *A/UX Local System Administration.*

Section 3, "Network Administration," describes the procedures for configuring your A/UX system for communication over a network. It supplements information in *A/UX Network System Administration,* which is available through the Apple Programmer's and Developer's Association (APDA).

Section 4, "Adding Extra Hard Disk SCs," explains how to attach an additional Apple Hard Disk SC to your system for use with either your Macintosh Operating System or your A/UX operating system. This section supplements information in your *A/UX Installation Guide.*

Section 5, "A/UX Toolbox," supplements information in *A/UX Toolbox: Macintosh ROM Interface.*

Section 6, "A/UX Kernel Messages," describes the error messages generated by the A/UX kernel.

Section 7, "Terminal Display Problems," provides information for configuring your display terminal after using a remote or non-A/UX terminal type.

Section 8, "Serial Port Difficulties," provides information regarding the Macintosh II device priorities of the SCC (serial) port interface and the IWM (floppy disk drive) interface.

Section 9, "RS-232 Cables," describes RS-232 cables used to connect DCE and DTE devices to the serial ports on a Macintosh II.

Section 10, "Manual Pages," contains a list of minor changes to printed manual pages and a set of replacement pages for printed manual pages.

Section 1

Programming Notes

This section contains a series of technical notes and warnings. It describes the origin of A/UX software, the differences between A/UX and other operating systems based on UNIX, and some details about this implementation. This section also describes the software-distribution disk and the handling of user address space.

What the A/UX system runs

The A/UX system is based on AT&T's UNIX System V Release 2 (V.2). It is a virtual-memory, time-sharing system with several additions, including

- TCP/IP Ethernet communication software
- 4.2 Berkeley Software Distribution (BSD) signals
- some 4.2 BSD system calls
- Sun Microsystems' NFS

A/UX resembles a V.2 system more closely than it resembles a 4.2 system. Many V.2 commands have different options from the corresponding 4.2 commands; please check your A/UX reference manuals. Also, many functions have different names on the two systems. (For example, the 4.2 BSD functions `index` and `rindex` are called `strchr` and `strrchr`, respectively, in V.2.)

Signals

A/UX supports two different versions of signals: the standard V.2 signals and the 4.2 reliable signals. To use 4.2 signals in a program, issue a call to the `set42sig` routine before issuing any calls to the `signal` function.

Do not attempt to mix the signal systems. Although you can mix 4.2 and V.2 signals using the `setcompat` and `getcompat` routines, this strategy is not recommended for a portable application. The signal-handler setting is not inherited across an `exec` call, although job control and process groups are available to shell programs. See the manual pages for `signal`(3), `set42sig`(2), `setcompat`(2), and `getcompat`(2) for more details.

Streams on the console

The console device (the video monitor) runs with a streams line discipline. This line discipline acts like the standard system V.2 line discipline, except that on the first open, you must push the line discipline module. If you do not, you will have only a raw terminal that provides no handling of line commands such as erase and kill. The standard UNIX programs, such as `getty` and `login`, push the line discipline module. However, if you disable all logins on a device and want to run your own application that expects standard line discipline handling, your application must push the line discipline module:

```
fd = open("/dev/console",O_RDWR);
      line_push(fd);
```

See the manual pages for `streams`(7), `termio`(7), `console`(7), `mouse`(7), `line_push`(3), and `stty(1)`.

VNODES and directories

A/UX runs a VNODE kernel. As a result, it is possible that a directory may not be a standard V.2 directory (in fact, it might not even be a UNIX directory). Apple recommends that any application that needs to read directories use the standard Berkeley directory access routines (supplied in the C library). Any program that needs to create a directory must use the `mkdir` system call. Do not use `mknod` to create a directory. See the manual pages for `directory`(3) and `mkdir`(2) for more information.

`rsh` and `remsh`

The standard V.2 system provides a command called `rsh`, a restricted version of the standard Bourne shell. The 4.2 networking command `rsh` has been renamed as `remsh`, for *remote shell*. See the manual pages for `sh`(1) and `remsh`(1) for more details.

Groups

A/UX runs 4.2-compatible groups by default. This means that any given user can be in more than one group at the same time. It also means that when files are created, the group ID of the file is the same as the group ID of the directory in which it is created, not the accounting group ID of the user. You can change this behavior with the `setcompat` and `getcompat` routines. See the manual sections for `groups`(1), `groups`(8), `setcompat`(2), and `getcompat`(2).

`termcap` and `terminfo`

A/UX provides two terminal capabilities data bases. The V.2 data base is a set of compiled capabilities files found under `/usr/lib/terminfo`. The other data base, `/etc/termcap`, is a text file that can be edited and that is used by BSD utilities ported to A/UX. In some future release of A/UX, these utilities will probably be converted to use `terminfo`, and the file `/etc/termcap` will be dropped.

When modifying terminal capabilities, keep in mind that A/UX utilities do not use both data bases. In particular, `tset` and `more` read `/etc/termcap`; `vi` uses `/etc/terminfo`.

Printing

A/UX uses the System V.2 suite of line printer programs (`lp`, `access`, `lpadmin`, `lpstat`, and so on). The Berkeley printing commands (`lpr`, `lpq`, and `lprm`) have been implemented as shell scripts. These implementations may not precisely duplicate the functionality of the original Berkeley utilities.

Kernel parameters

On a system with more than 2 megabytes (MB) of memory, you might be able to improve system performance by altering some kernel parameters. See the discussion of resetting kernel parameters in the "System Administration" section of these release notes.

Access to the video monitor and mouse

See the `console`(7) and `mouse`(7) manual pages for information about access to the video monitor and mouse.

Big C compiler

The directory `/usr/lib/big` contains versions of the C compiler, assembler, and link editor for compiling large programs. These tools are the same as their standard counterparts, except that their symbol tables are very large. Use these large versions of the C tools if you get a symbol table overflow error while compiling your program.

To invoke the big C compiler, add this flag to your `cc` command line:

```
cc -B /usr/lib/big/ -o filename filename.c
```

See the `cc`(1) manual page in *A/UX Command Reference* for more information. Use of the big C compiler is not recommended on A/UX systems with 2 MB of memory.

Public domain utilities

This distribution includes source code for several useful public domain programs: `emacs` (a popular text editor), `kermit` (a file transfer protocol), `compress` (a set of file compression utilities that use a faster and better algorithm than `pack` and `compact`), and `pcnfsd` (a utility for using the Network File System with an IBM PC). This source code is located in the directory `/usr/src`.

Warning
These public domain utilities are provided as a courtesy to A/UX users, and *are not supported* by Apple Computer, Inc.

These programs are not compiled. Manual pages, if they exist, are unformatted and are located with the source code. An exception is `kermit`, which exists in binary form in `/usr/bin`. To see the manual page for `kermit`, enter the command `man kermit`.

If you need more space on your disk, you can delete or compress some or all of this source code.

Layout of the Apple Hard Disk 80SC

The Apple Hard Disk 80SC, preloaded with A/UX, is divided into nine partitions. The disk contains both A/UX and Macintosh Operating System (OS) partitions. The size of a block is 512 bytes.

Table 1-1 shows the block definitions.

Table 1-1
Block definitions

Start	End	Blocks	Description
0	0	1	Hard disk block zero
1	63	63	Apple partition map (63 entries)
64	95	32	Macintosh OS HD 80SC disk driver
96	127	32	Reserved for HD 80SC disk driver expansion
128	6,271	6,144	A/UX eschatology (autorecovery) file system 1
16,272	117,455	111,184	A/UX root (/) and user (/usr) file system
111,185	146,127	28,672	A/UX swap space
146,128	150,223	4,096	Macintosh OS HFS (A/UX booter)
150,224	156,367	6,144	A/UX eschatology (autorecovery) file system 2
156,368	156,369	2	Free space and padding

If you want to remove some disk contents to make more room for user data, see the "System Administration" section of these release notes.

If you want to add an additional hard disk to hold user data, see the section "Adding Extra Hard Disk SCs" in these release notes.

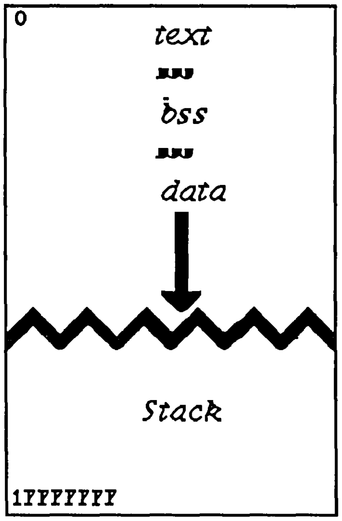
Address space

Figure 1-1 illustrates the organization of user process address space. Note that the total available address space is 512 MB.

Figure 1-2 shows the kernel address map, which you need if you are writing an A/UX device driver.

SIZES	
Page size	4K
Pages per segment	256
Segment size	1M
Maximum segments	512
User address space	512M
Kernel address space	4G

USER ADDRESS SPACE



The User Virtual Address

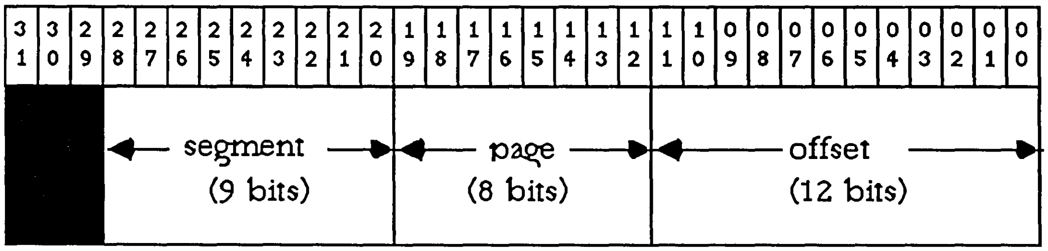


Figure 1-1
Organization of user address space

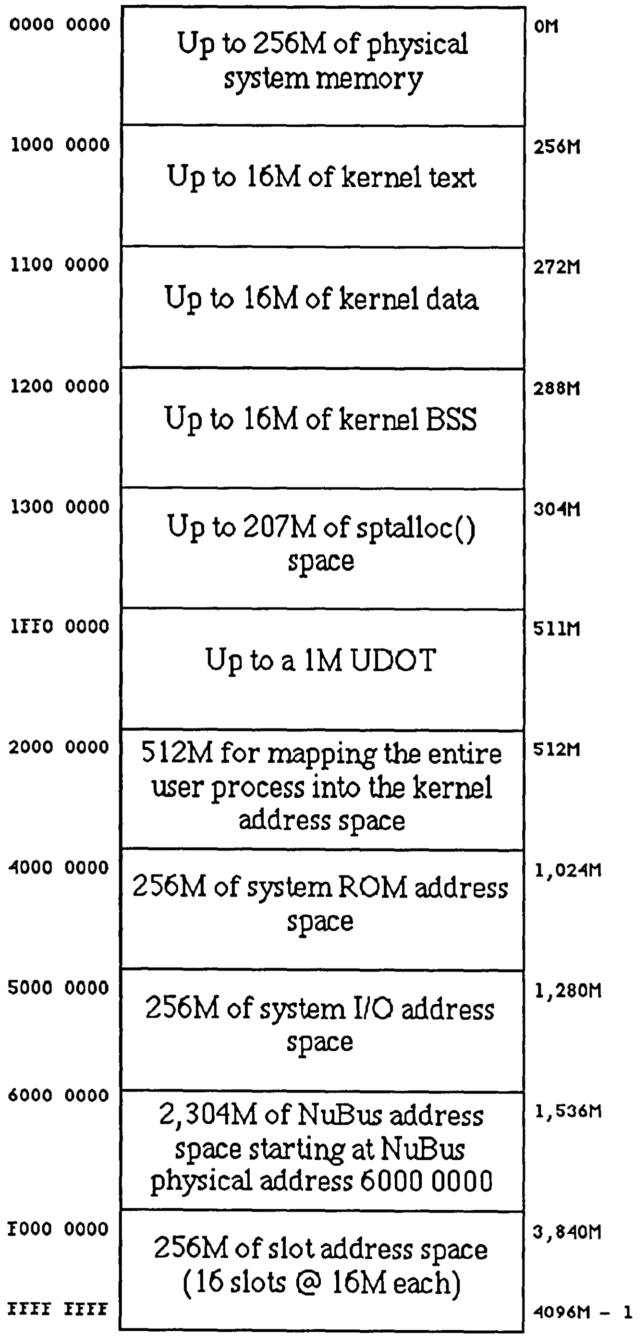

0000 0000
Up to 256M of physical system memory
0M
1000 0000
Up to 16M of kernel text
256M
1100 0000
Up to 16M of kernel data
272M
1200 0000
Up to 16M of kernel BSS
288M
1300 0000
Up to 207M of sptalloc() space
304M
1FF0 0000
Up to a 1M UDOT
511M
2000 0000
512M for mapping the entire user process into the kernel address space
512M
4000 0000
256M of system ROM address space
1,024M
5000 0000
256M of system I/O address space
1,280M
6000 0000
2,304M of NuBus address space starting at NuBus physical address 6000 0000
1,536M
F000 0000
256M of slot address space (16 slots @ 16M each)
3,840M
FFFF FFFF
4096M - 1

Figure 1-2
Kernel address map

Device drivers

At first release, the A/UX kernel includes these device drivers:

- Apple Desktop Bus™ (ADB). This supports the keyboard and mouse. Both a standard ASCII keyboard interface and a key-up/key-down interface are available. Mouse events are available.
- Video driver and console emulator. The console terminal emulator supports a subset of VT100 terminal features.
- Floppy driver. Both single-sided (400K) and double-sided (800K) disks are supported, although only 800K drives are normally available on the Macintosh II hardware. Both 400K and 800K disks can be read, written, ejected, and formatted.
- SCSI manager and SCSI disk driver. These support the Apple Hard Disk 80SC drive.
- Serial Communications Controller (SCC) driver.
- NV ("Parameter") RAM and time-of-day clock driver.
- Ethernet driver, with full TCP/IP and NFS support.
- The A/UX Toolbox driver (see the section "A/UX Toolbox" in these release notes).

Section 2

System Administration

This section supplements *A/UX Local System Administration.* It covers five system administration issues:

- reclaiming disk space
- booting into multiuser mode
- changing kernel parameters
- using autoconfiguration
- using autorecovery

For information on network administration issues, see the section "Network Administration" in these release notes.

Reclaiming disk space

The A/UX release includes all the files needed to run A/UX locally, plus the source files for an array of public domain software. Single-disk systems do not have much room available for user data.

You can make more room on your disk by removing software that you do not use. For example, you can delete or compress the public domain software or delete the on-line manual pages. Always back up files before deleting them from the disk.

If you have more than one Macintosh II on a network, you can store shared resources such as manual pages on one Macintosh II, and remove those files from other computers on the same network. Both the server machine and the client machines must be configured for NFS.

To set up one Macintosh II as a manual-pages server, for example, follow these steps:

1. On the server, make the `/usr/catman` directory exportable by using this process:
   - ☐ Become the superuser by entering the command
     ```
     su
     ```
     password
   - ☐ Edit the file `/etc/exports`, adding entries for `/` and `/usr/catman`.
2. Set up the client computers to mount the `/usr/catman` directory remotely by using this process:
   - ☐ Change to the `/usr` directory with the command
     ```
     cd /usr
     ```
   - ☐ Become the superuser by entering the command
     ```
     su
     ```
     password
   - ☐ Remove the man page source files by entering the commands
     ```
     rm -r catman
     mkdir catman
     ```
   - ☐ Edit the file `/etc/fstab` to add this line:
     ```
     server_name:/usr/catman /usr/catman nfs soft,ro,bg 0 0
     ```
   - ☐ Mount `/usr/catman`:
     ```
     mount /usr/catman
     ```

Booting into multiuser mode

As shipped, A/UX comes up in single-user mode. You must respond to a series of prompts, and then explicitly enter multiuser mode. This sequence is useful when you are first working with A/UX, looking around the system, and setting things up. (See your *A/UX Installation Guide* for more information.)

After you have set up your system, you might wish to bypass single-user mode and the prompt for the file system check.

To cause A/UX to come up automatically into multiuser mode, edit the file `/etc/inittab`. Change the line

```
is:s:initdefault
```

to

```
is:2:initdefault
```

To check the file systems automatically after a 5-second wait, without prompting, edit the files `/etc/bcheckrc` and `/etc/sysinitrc`. In both files, change the line

```
reply="`/etc/query`"
```

to

```
reply="`/etc/query -t 5`"
```

When you next boot the system, it should come up in multiuser mode, displaying this prompt:

```
Apple Computer, Inc.  A/UX

login:
```

Changing kernel parameters

The A/UX software as shipped from the factory is configured for a system with 2 MB of memory. If your system contains more memory, you might be able to improve performance by changing some kernel parameters. Do not change any parameters on a system with 2 MB of memory.

On a system with 4 MB or 5 MB, bring A/UX up into single-user mode, become the superuser, and enter the following commands (enter CONTROL-d by holding down the CONTROL key while pressing the d key):

```
kconfig -n /unix
NBUF=1000
NINODE=200
NFILE=200
CONTROL-d
sync
sync
sync
reboot
```

On a system with 8 MB, bring A/UX up into single-user mode, become the superuser, and enter the following commands:

```
kconfig -n /unix
NBUF=2500
NINODE=600
NFILE=400
CONTROL-d
sync
sync
sync
reboot
```

Autoconfiguration

Autoconfiguration is a utility that runs automatically at boot time and checks the consistency between the hardware that is attached to your Macintosh II and the information in the kernel about attached devices. Autoconfiguration allows you to change your hardware configuration without resetting DIP switches or manually rebuilding the kernel. Autoconfiguration shields you from detailed knowledge of the interface between the hardware and the kernel, and protects against operating-system problems caused by mismatched hardware and software.

How autoconfiguration works

The A/UX kernel contains information about the devices installed in each slot. Whenever A/UX is booted, autoconfiguration checks this information against the cards installed in the slots. If the kernel contains drivers for hardware that is absent, or if known hardware is attached through a different slot than the kernel expects, the autoconfiguration program makes the appropriate changes to the kernel and then reboots the system.

Adding new devices

Autoconfiguration does not automatically add to the kernel any information about new devices. Autoconfiguration can work only with the device information already in the A/UX kernel. The system will not attempt to use a device until the kernel is configured for that device.

When you add a new device, you must run the installation scripts provided with that device. These scripts use autoconfiguration to add the appropriate information to the kernel. You cannot use the device until after you reboot the system, at which time autoconfiguration checks that the hardware setup matches the kernel's information.

If autoconfiguration detects a device for which the kernel has no driver, it prints a warning message. For example, when autoconfiguration locates the video card, it prints this message:

```
Warning cannot find drivers for device ID 5 Version 7
```

You can safely ignore this warning.

Where autoconfiguration is documented

Autoconfiguration is described on the `autoconfig`(1M) manual page in *A/UX System Administrator's Reference.*

Autorecovery

Autorecovery is a set of programs designed to ensure that an A/UX system can be brought to multiuser mode, and that it will then be able to operate as part of a network. Before the system is booted, autorecovery identifies and compensates for bad disk blocks, file-system inconsistencies, and missing or damaged files. Autorecovery is designed to protect you from sudden, catastrophic loss of data and to minimize the need for a technical expert to diagnose and repair system problems.

Autorecovery 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 autorecovery's operation is automatic, you must perform some administration tasks to keep autorecovery running smoothly. This section describes the administration tasks.

Overview

The terms *autorecovery* and *eschatology* are used to refer to the same A/UX feature. *Autorecovery* refers to the procedure that checks and repairs the A/UX file systems and to the special disk file systems used by this procedure. *Eschatology* is used in filenames and command names and in the arguments to commands.

Two areas on the A/UX disk are reserved for autorecovery. Each 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 one of these systems to the A/UX root file system.

Autorecovery uses a list of key system files known as the *configuration master list,* or CML. The CML appears in the A/UX root file system in the file `/etc/eschatology/init2files`. A copy of this file appears in each autorecovery file system.

At boot time, autorecovery verifies the physical condition of the disk and then checks each file in the CML. Autorecovery 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 upon the CML to perform its duties, 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 systems, using the autorecovery utilities `eu` and `escher` (discussed later). Autorecovery depends upon a conscientious system administrator to keep the CML and autorecovery file systems updated.

Using autorecovery

Autorecovery is run from the standalone shell (`sash`) when the system is booted. (See *A/UX Local System Administration* and the manual page for `sash`(8) in *A/UX System Administrator's Reference* for information on `sash`.) You can either run autorecovery manually each time you boot or set it up to be run automatically.

To run autorecovery manually from `sash`, enter the command

```
esch -v -b
```

For a detailed explanation of the `esch` command and its options, see `esch`(8) in *A/UX System Administrator's Reference.*

To set up the system to run autorecovery each time you boot, follow these steps:

1. Select the Cancel option from the `sash` window.
2. Select the Booting option from the Preferences menu.
3. Enter this command in the autorecovery box:
   ```
   esch -v -b
   ```
4. Click the OK button.

To suppress the automatic running of autorecovery, follow the same steps, but replace the `esch` command in the autorecovery box with

```
echo no autorecovery
```

How autorecovery works

Autorecovery proceeds through several phases. First, autorecovery 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 systems are suspect, autorecovery cannot use them to restore damaged or missing files in the A/UX root file system. If autorecovery detects that an autorecovery file system was being updated when the system failed, it does not use that file system. If both autorecovery file systems are suspect, autorecovery does not continue.

After checking its own file systems, autorecovery checks each A/UX file system individually, verifying that all blocks are readable. It marks bad blocks so they will not be used.

Autorecovery then checks the A/UX root file system and each autorecovery file system for consistency, by using a version of `fsck`. Autorecovery 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 are lost and must be restored from backups.

After all file systems have been checked, autorecovery verifies the A/UX key system files listed in the CML, checking the attributes of each file against the rules specified in the CML. If autorecovery finds inconsistencies, it corrects the file, if possible, or replaces it with a copy from the autorecovery file systems.

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 *A/UX Local System Administration* for a description of the file system checking routine (`fsck`).

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 `escher`, `eu`, and `eupdate` utilities, described in this section.

The utilities `escher` and `eu` perform similar functions: both update the CML and the autorecovery file systems, but they use slightly different procedures. The utility `eupdate` copies the system files typically updated by autoconfiguration to the autorecovery file systems and updates the CML entries for these files. Each utility is described next and in *A/UX System Administrator's Reference.*

`eu` utility

The `eu` utility updates the CML and copies a specified file to the autorecovery file systems. The `eu` utility operates on only one file at a time and, like `escher`, requires a fully qualified pathname for the file to be copied. To run `eu`, enter the command

`eu` *pathname*

where *pathname* is the complete pathname of the file to be copied. A complete 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. The escher utility updates the CML only if it does not already contain an entry for the file to be copied.

You must be the superuser to run eu.

escher utility

The escher utility also adds new entries to the CML and copies the file to the autorecovery file systems. To run escher, enter the command

```
escher pathname
```

where *pathname* is the complete pathname of the file to be added. A complete pathname always begins with a slash (/).

The escher utility examines the specified file and adds information about the file to the CML. As discussed in the manual 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 the eu utility. When the CML entry has been created, escher copies the file to both autorecovery file systems.

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 systems. For each file that escher finds, it prompts you to decide whether the file should be copied to each of the two autorecovery file systems. When escher completes, the autorecovery file systems are current.

You must be the superuser to run escher.

eupdate utility

The eupdate utility updates the CML entries for the files typically modified by autoconfiguration. To run eupdate, type the command

```
eupdate
```

The A/UX kernel (/unix) and other files necessary for multiuser network operation are copied to the autorecovery file systems. The CML entries for these files are also updated.

You must be the superuser to run eupdate.

Administration guidelines

To keep your autorecovery file systems current, follow these guidelines:

- Run `escher -m` often, to obtain a list of files that may need to be updated in the autorecovery file systems. 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. Autorecovery cannot recover files that do not appear in the CML.
- Run `eu` or `eupdate` as soon as key system files change. For example, if the `autoconfig` command is run to build a new kernel, `eupdate` should be run as soon as the operation of the new kernel is verified.

The `escher` and `eu` utilities may be used to add entries to the CML. Currently, no utility is available to delete CML entries. Use a text editor for this task. Consult the manual page for `cml` before attempting to update the CML manually.

Troubleshooting

This section discusses problems that occasionally occur when running autorecovery. Although autorecovery is fairly robust, it errs on the side of caution: autorecovery will not replace damaged or missing files if the copies on the autorecovery file systems or the autorecovery file systems themselves are 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 systems. Manual intervention should be kept to a minimum. The procedures recommended here use the `pname` utility. Users unfamiliar with this utility should consult the manual page for `pname`(1M) 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.

Symptom: One or both autorecovery file systems have inconsistent mount times.

Symptom: Autorecovery fails during boot.

Message: `Warning. Inconsistent mount times in bzb.`

Message:
```
esch: no consistent type FSTEFS
BZB (ES_BZBS_FSTEFS) [error occurred in Block Zero
Block module]
```

These messages may appear when autorecovery is run from the `sash` partition. Occasionally, autorecovery may be unable to restore damaged or missing files. Usually, this result occurs because autorecovery file systems were left mounted when the system was halted. After detecting this condition, autorecovery will not restore files from any autorecovery file systems that were left mounted when the system was interrupted. Autorecovery will not use these file systems because the integrity of the data is not guaranteed. If both autorecovery file systems were mounted when the system was halted, autorecovery cannot proceed.

The most obvious sign that autorecovery has been interrupted is the message just listed. Autorecovery verifies that the unmount time is later than the mount time for each autorecovery file system. If an autorecovery file system has been left mounted, this test will fail and will generate the warning message. The autorecovery file systems that were left mounted when the system was halted must be checked for integrity. Use the following procedure to check the offending file system or systems.

1. Launch `/unix` from `sash`. Do not run autorecovery.
2. At the `sash` prompt, enter the command
   ```
   launch /unix
   ```
3. When the system prompts you to check the root file system, type `y`.
4. Bring the system to single-user mode.
5. At the root prompt, enter the command
   ```
   pname -s3 "Eschatology 1"
   ```
6. At the root prompt, enter the command
   ```
   fsck /dev/dsk/c0d0s3
   ```
   The system will check one autorecovery file system for consistency.
7. At the root prompt, enter the command
   ```
   mount /dev/dsk/c0d0s3 /mnt
   ```
8. Enter the command
   ```
   umount /mnt
   ```
9. Enter the command
   ```
   pname -u /dev/dsk/c0d0s3
   ```
10. Enter the command
    ```
    pname -s3 "Eschatology 2"
    ```
11. Enter the command
    ```
    fsck /dev/dsk/c0d0s3
    ```
    The system will check the second autorecovery file system for consistency.

12. Enter the command

```
mount /dev/dsk/c0d0s3 /mnt
```

13. Enter the command

```
umount /mnt
```

14. Enter the command

```
name -u /dev/dsk/c0d0s3
```

15. Enter the command

```
sync;sync;reboot
```

The system will restart.

Autorecovery should now be operational.

Symptom: `escher` or eu will not run.

Message: `Can't lock cml file.`

Message: `eu: Can't lock the fcml. Try again later.`

These messages are produced when both `escher` and eu are preparing to update the CML at the same time. The two utilities cannot be run at once, 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 `escher` or eu is running, the file `/etc/eschatology/FCML.lock` may be present. After verifying that no processes are attempting to update the CML, you may remove this file.

Symptom: `escher` will not run.

Message: `/dev/dsk/cXXdYYsZZ previously pnamed`

Message: `/dev/rdsk/cXXdYYsZZ previously pnamed`

These messages are produced when at least one device is still associated with an autorecovery file system. To remove the association, use the pname command by following these steps:

1. Enter the command

```
pname
```

The system will produce 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/c0d0s4: "Eschatology 2" "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, with the example just given, you enter the commands

```
pname -u /dev/dsk/c0d0s3
pname -u /dev/dsk/c0d0s4
```

3. Verify that no devices are currently associated with autorecovery file systems by entering

   ```
   pname
   ```

   again. 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 complete normally.

Symptom: "Replaced" files missing after autorecovery completes.

Message: `replacing` *filename*

Message: *filename* `was not replaceable.`

These messages are produced when autorecovery is run at boot time. When autorecovery determines that a file is invalid, it attempts to replace it with a valid copy from the autorecovery file systems. If the autorecovery file systems do 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 systems but fail the rules specified in the CML. Consistent use of the `eupdate`, `eu`, and `escher` utilities should prevent this problem from occurring.

Autorecovery removes invalid files before attempting to replace them by design. Since autorecovery is concerned only with key system files, damage to these files could pose a security risk, especially in a network environment. For this reason, autorecovery removes files identified as invalid, whether or not a suitable replacement is available.

Symptom: Some files cannot be copied to autorecovery file systems.

Message: `Ignoring` *filename* `- cmlfile entry incorrect`

When run interactively, `escher` produces this message. The `escher` utility will complain 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 systems.

Section 3

Network Administration

This section explains how to configure your system for networking, using either TCP/IP only (BNET) or TCP/IP with the Network File System (NFS). The section also discusses how to set up your system to be a Yellow Pages client and how to bring up `sendmail`.

EtherTalk card installation notes

Before configuring A/UX, you should have an EtherTalk™ card installed. Complete installation instructions are provided when you purchase the card. Note, however, that you should examine the back end of your EtherTalk card before installation. On many cards, the R33 resistor may come into contact with the ferrite bead. This situation will cause shorting. Gently bend the resistor away from the ferrite bead, to prevent possible shorts. Figure 3-1 shows this process.

Also, be especially careful if you are using thin Ethernet cable and must reposition the jumper on your card. If you have the shorting jumper block placed over the wrong two columns, you will select the wrong cable type. The EtherTalk card will try to send and receive on the wrong cable connector, and your Ethernet will not work.

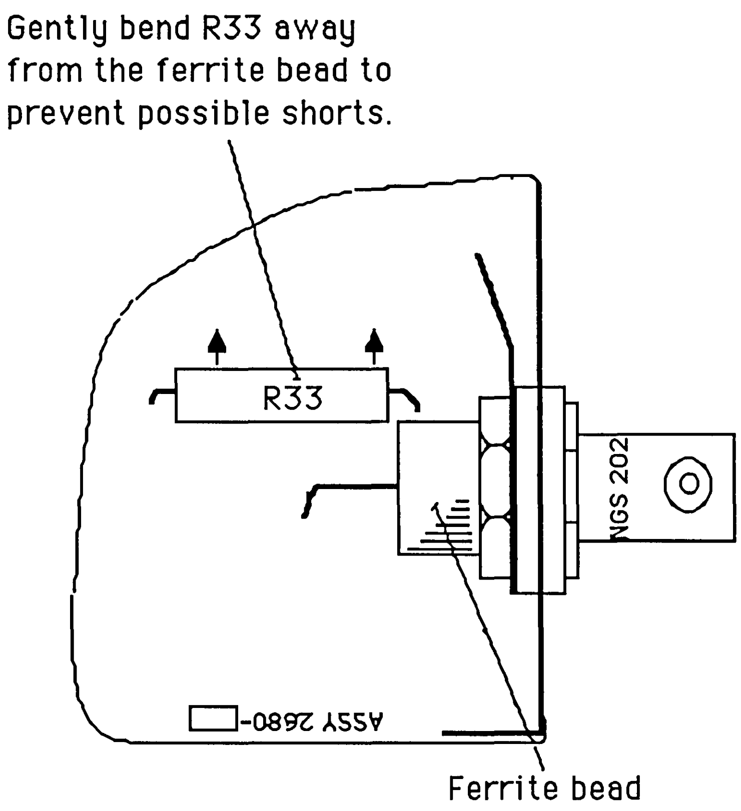

Gently bend R33 away from the ferrite bead to prevent possible shorts.
R33
NGS 202
ASSY 2680-
Ferrite bead

Figure 3-1
Preventing shorting of EtherTalk card

If the jumper is over the wrong two columns, the Ethernet software driver in the A/UX kernel usually outputs an error message. The error message given commonly is

```
ae0 transmitter frozen -- resetting
```

Warning

If you are using thin Ethernet cable, be sure that your T-shaped BNC connector is terminated. If your machine is at the end of the cable, there must be a termination cap on the leg of the connector that does not have a cable attached.

Configuring A/UX for TCP/IP (BNET) use (no NFS)

To configure your system to communicate over TCP/IP but not NFS, follow these steps:

1. Log in as `root`, and bring the system down to single-user mode.
2. Edit the file `/etc/inittab` to change the appropriate lines. In the following illustration of the file, the changed words appear in bold:

```
sy::sysinit:/etc/sysinitrc </dev/syscon >/dev/syscon 2>&1
is:s:initdefault:
bl::bootwait:/etc/bcheckrc </dev/syscon >/dev/syscon 2>&1
bc::bootwait:/etc/brc </dev/syscon >/dev/syscon 2>&1
rc::wait:/etc/rc 1>/dev/syscon 2>&1
sl::wait:(rm -f /dev/syscon;ln /dev/systty /dev/syscon;) 1>/dev/systty
  2>&1
pf::powerfail:/etc/powerfail 1>/dev/syscon 2>&1
er:2:once:/usr/lib/errdemon
cr:2:once:/etc/cron </dev/syscon >/dev/syscon 2>&1
lp:2:once:/usr/lib/lpsched >/dev/syscon 2>&1
co::respawn:/etc/getty console co_9600
tb0:2:off:/etc/toolboxdaemon > /dev/syscon 2>&1
nfs0:2:wait:/etc/portmap
nfs1:2:off:/etc/ypserv
nfs2:2:off:/etc/ypbind
nfs3:2:off:/etc/nfsd 4
nfs4:2:off:/etc/biod 4
nfs8:2:off:/etc/mount -at nfs > /dev/syscon 2>&1
net4:2:off:/etc/in.routed
net5:2:once:/usr/etc/in.rwhod
net8:2:off:/usr/lib/sendmail -bd -q30m
net9:2:respawn:/etc/inetd
net0:2:off:/etc/named /etc/named.boot
00:2:off:/etc/getty tty0 at_9600
01:2:off:/etc/getty tty1 at_9600
```

If you will be using a network with a gateway, also change the line

```
net4:2:off:/etc/in.routed
```

to

```
net4:2:wait:/etc/in.routed
```

3. Enter the following commands to configure a BNET kernel:

   ```
   /etc/newunix bnet
   /etc/autoconfig -u -v -o /unix -S /etc/startup
   ```

   The system responds with many lines of information.

   ```
   sync
   sync
   reboot
   ```

4. Bring the system back up, but do not run autorecovery.
5. After the reboot, the system prompts you for internet addresses and other related pieces of information. See "Selecting an Internet Address," below, for information on the way to answer these questions.
6. Verify the operation of the new kernel and modified files. Run `/etc/eupdate` to update the autorecovery file systems correctly.

Configuring A/UX for TCP/IP and NFS

To configure your system to run TCP/IP and to be an NFS client or server, follow these steps:

1. Log in as `root` and bring the system down into single-user mode.
2. Edit the file `/etc/inittab` as described previously in "Configuring A/UX for TCP/IP (BNET) Use (No NFS)."
3. Make the following additional changes to `/etc/inittab`:

   If you plan to be an NFS server, change the line

   ```
   nfs3:2:off:/etc/nfsd 4
   ```

   to

   ```
   nfs3:2:wait:/etc/nfsd 4
   ```

   If you plan to be an NFS client, change the lines

   ```
   nfs4:2:off:/etc/biod 4
   nfs8:2:off:/etc/mount -at nfs > /dev/syscon 2>&1
   ```

   to

   ```
   nfs4:2:wait:/etc/biod 4
   nfs8:2:once:/etc/mount -at nfs > /dev/syscon 2>&1
   ```

   Regardless of whether you are a server or a client, if you will be using a network with a gateway, also change the line

   ```
   net4:2:off:/etc/in.routed
   ```

   to

   ```
   net4:2:wait:/etc/in.routed
   ```

4. Enter the following commands:

```
/etc/newunix nfs
/etc/autoconfig -u -v -o /unix -S /etc/startup
```

The system responds with many lines of information.

```
sync
sync
reboot
```

5. Bring the system back up, but do not run autorecovery.
6. After the reboot, the system prompts you for internet addresses and other related pieces of information. See "Selecting an Internet Address," below, for information on how to answer these questions.
7. Verify the operation of the new kernel and modified files. Run `/etc/eupdate` to update the autorecovery file systems correctly.

Configuring A/UX as a Yellow Pages client with either BNET or NFS

To configure A/UX as a Yellow Pages client, follow these steps:

1. Edit the file `/etc/inittab` and change the line

```
nfs2:2:off:/etc/ypbind
```

to

```
nfs2:2:wait:/etc/ypbind
```

2. Add the following line to the end of the `/etc/passwd` file:

```
+::0:0:::
```

3. Add the following line to the end of the `/etc/group` file:

```
+:::
```

Selecting an internet address

This section explains how to select the various names and numbers needed by A/UX to run on a network. You set these names and numbers by responding to questions asked during the configuration of A/UX for network use, as described earlier.

Host name

You must name your A/UX system. Every A/UX system on the Ethernet must have a name by which it is known on the network. This name is the *host name.* A host name should be from 1 to 8 characters long.

You should have set your host name when you first brought your system up. (See your *A/UX Installation Guide* for details.) If you wish to change your host name now, edit the file `/etc/HOSTNAME`. (See the manual page for `HOSTNAME`(4) in *A/UX Programmer's Reference.*) In this example, the host name is *golden.*

Internet address

The *internet address* is a number that is unique to each system on the network. If you think of the host name as being like your name, such as *John Smith,* then the internet address could be like your home address, *1234 Main Street.*

An internet address is 32 bits, or 4 bytes, in length. The 4 bytes are separated by periods (for example, 192.33.20.34). There are three classes of internet address, known as A, B, and C. These classes differ in the number of bytes taken up by the network number and the number of bytes available for the host numbers. This section describes class C addresses only.

In a class C address, the first 3 bytes constitute the network number (in the example just given, 192.33.20), which is shared by all machines on a given network. The remainder of the address (34) is the host number, which is unique for each machine on the network.

You must inform your system of its internet address, as well as its site-specific broadcast address and netmask (if any). Your broadcast address is formed like any other internet address, except that the host number is either 0 or –1 (entered as 0 or 255, respectively) for a class C address.

Warning
You should check with your system administrator or network administrator to obtain an internet address for your machine. If you have no network administrator, you should use the network number given in the next example. Be sure that all machines on your network share the same network number but have unique host numbers.

These instructions apply only to networks that do not have a network administrator and that use these instructions for all systems on the network. If you wish to install your system in a network where not everyone is following these instructions, you must get your internet address from a system administrator or network administrator. If you think you may connect your network to another network in the future, obtain an official, registered network number by calling

Hostmaster
(800) 235-3155

Example

The following example assigns a class C network number and host number to the machine with host name *golden*. The example uses the default netmask, implying a non-subnetted environment. The range of possible host numbers for a class C address is 1 to 254. Host number 34 is assigned to *golden*. The network number for this class C network is 192.33.20.

Before assigning an internet address to your system, you must reconfigure the kernel for either TCP/IP–**only (BNET) or NFS support, as described in the earlier parts of this section in "Configuring A/UX for TCP/IP (BNET) Use (No NFS)" or in "Configuring A/UX for TCP/IP and NFS." When the new A/UX kernel finds an EtherTalk card for the first time, it prompts for the internet address. The dialog proceeds as follows (user responses are in bold type):

```
lo0: 127.0.0.1 netmask ff000000 flags=421<UP,NOTRAILERS,LOOPBACK>
   1 Ethernet card(s) installed
ae0: Please enter an Internet address: 192.33.20.34 <RETURN>
ae0: Please enter an Internet Broadcast address: 192.33.20.255 <RETURN>
ae0: Please enter a netmask [none]: <RETURN>
ae0: hostname is golden with IP address of 192.33.20.34 broadcasting with
   192.33.20.255
ae0: 192.33.20.34 netmask ffffff00 flags=63<UP,BROADCAST,NOTRAILERS,RUNNING>
broadcast: 192.33.20.255
```

If you wish to change these values at a future date, edit the file /etc/NETADDRS. (See the manual page for NETADDRS(4) in *A/UX Programmer's Reference*.)

For more information on network classes, as well as netmasks and subnetted environments, see your network administrator or consult the appropriate RFCs available from

The NIC (Network Information Center)
333 Ravenswood
SRI International
Menlo Park, CA 94025
(415) 326-6200

For additional information on setting up and administering a networked Macintosh II, see *A/UX Network System Administration,* available through the Apple Programmer's and Developer's Association (APDA):

Apple Programmer's and Developer's Association
290 SW 43rd Street
Renton, WA 98055
(206) 251-6548
AppleLink: APDA
MCI: 321-7449
CompuServe# 73527,27

Configuring `sendmail`

This section describes how to bring up `sendmail`. More complete information appears in *A/UX Network System Administration,* available through the Apple Programmer's and Developer's Association (APDA).

A/UX is shipped with all the normally needed aliases, such as `postmaster`. If you wish to modify the aliases, edit the file `/usr/lib/aliases`. If you change the aliases, you must run `/usr/ucb/newaliases` before the changes will be visible to `sendmail`. For more information, see the manual page for `aliases`(4).

A/UX is also shipped with a default configuration file (`/usr/lib/sendmail.cf`), which is appropriate for most sites. It should, however, be frozen so that `sendmail` will start up faster. You can freeze the file by executing

```
/usr/lib/sendmail -bz
```

If you desire to change the default configuration, you can find more information in *A/UX System Administrator's Reference* under `sendmail`(1M).

To have `sendmail` run as a standard process when the system boots, change

```
net8:2:off:/usr/lib/sendmail -bd -q30m
```

in `/etc/inittab` to

net8:2:**once**:/usr/lib/sendmail -bd -q30m

You may also wish to reduce the interval for processing the messages in the queue. This value is currently set at 30 minutes. Change the value shown in bold here:

net8:2:once:/usr/lib/sendmail -bd -q**30**m

Section 4

Adding Extra Hard Disk SCs

Apple Computer makes three external SCSI hard disks for increasing your system's mass storage capability: the HD 20SC, the HD 40SC, and the HD 80SC. You can use any of these disks for mounting an A/UX file system or for storing files and applications accessible from the Macintosh Operating System. In addition to the Hard Disk SC that contains A/UX, you can add up to six more hard disks by attaching them to the SCSI port on your Macintosh II computer.

Chapter 8 of the *A/UX Installation Guide* refers to several guides that can help you add disk storage to your system. This release note has been prepared since the *A/UX Installation Guide* went to press to offer you comprehensive instructions for attaching and configuring Apple Hard Disk SCs to an A/UX system.

As long as you're adding Hard Disk SCs from Apple Computer to your system, this document alone should be sufficient in providing you with all the installation and troubleshooting information you need. For additional setup information, you might still wish to refer to the *Apple SCSI Cable System* and to the owner's guide for your Hard Disk SC.

To add an Apple Hard Disk SC to your system, you need the following components:

- □ an Apple Hard Disk SC
- □ the *Hard Disk SC Setup* disk (shipped with each Apple Hard Disk SC)
- □ a Hard Disk SC power cord
- □ an Apple SCSI System Cable
- □ an Apple SCSI Cable Terminator

If you're adding more than one external hard disk to your system, you need a SCSI Peripheral Interface Cable for each additional Hard Disk SC. Figure 4-1 identifies these items for you.

❖ *Note:* Do not attach your external Hard Disk SC to your computer or plug the power cord into the Hard Disk SC until you're instructed to do so.

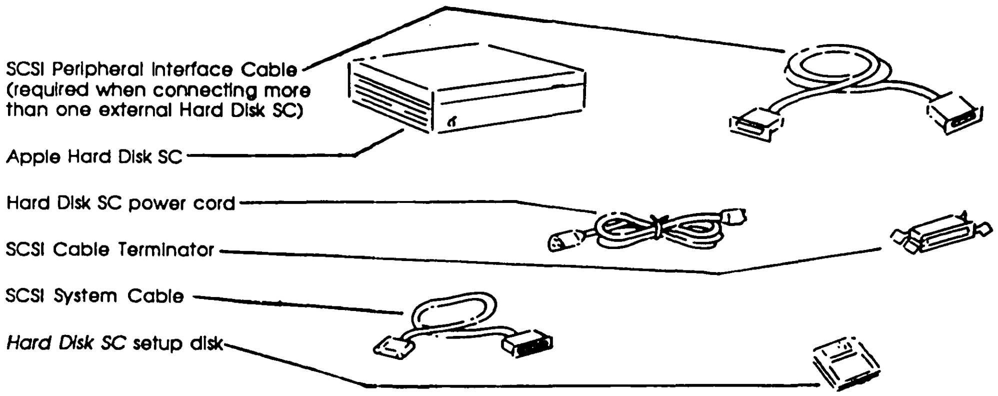

SCSI Peripheral Interface Cable (required when connecting more than one external Hard Disk SC)
Apple Hard Disk SC
Hard Disk SC power cord
SCSI Cable Terminator
SCSI System Cable
Hard Disk SC setup disk

Figure 4-1
Hard Disk SC and SCSI components

Attaching Hard Disk SC hardware

This section describes how to connect external Hard Disk SCs, their cables, and terminating resistors (in the form of SCSI Cable Terminators) to the Macintosh II computer.

1. Shut down A/UX, and turn off the Macintosh II and any attached SCSI devices.

The procedures for shutting down and switching off the Macintosh II are explained in detail in Chapter 6 of the *A/UX Installation Guide* and Chapter 2 of *A/UX Local System Administration.* In brief, these procedures are as follows:

- □ Log in as `root`.
- □ Use the `shutdown` command to return to single-user mode.
- □ Enter the `sync` command at least twice.
- □ Enter the `powerdown` command to turn off the Macintosh II.
- □ If you have any SCSI devices connected to your system, switch them off now.

2. Touch any one of the metal connectors on the back of your computer.

Doing this discharges any static electricity that may be on your body or clothes.

3. **Decide upon a unique SCSI ID number for the additional Hard Disk SC. (Normally, use 5 for the first external hard disk and 6 for the second.)**

   The SCSI standard lets you connect up to seven peripheral devices to form a SCSI chain. In order to establish a communication path, each device must have a unique SCSI ID number, which designates the priority and logical location of a device on the SCSI chain. The SCSI ID number identifies each peripheral device regardless of its physical location in your SCSI chain.

   You must assign a unique SCSI ID number to each Hard Disk SC you add to your system. Your Macintosh II is always assigned ID number 7. By convention, the internal hard disk uses SCSI ID number 0, the first external hard disk uses SCSI ID number 5, and the second external hard disk uses SCSI ID number 6. For any additional peripheral devices, use the remaining SCSI ID numbers to organize your SCSI chain.

4. **Set the SCSI ID for the additional Hard Disk SC.**

   After you have chosen a unique SCSI ID, set your Hard Disk SC to that number. (See Figure 4-2 for the location of the SCSI ID number switch and SCSI ID number indicator.)

   - □ Make sure the power cord is *not* connected to the additional Hard Disk SC so that it cannot possibly be turned on.
   - □ Insert the point of a pin or straightened paper clip into the SCSI ID number switch on the back of the Hard Disk SC.
   - □ Push gently. The number increases.
   - □ If you go past the number you want, keep pushing until the number cycles around again.

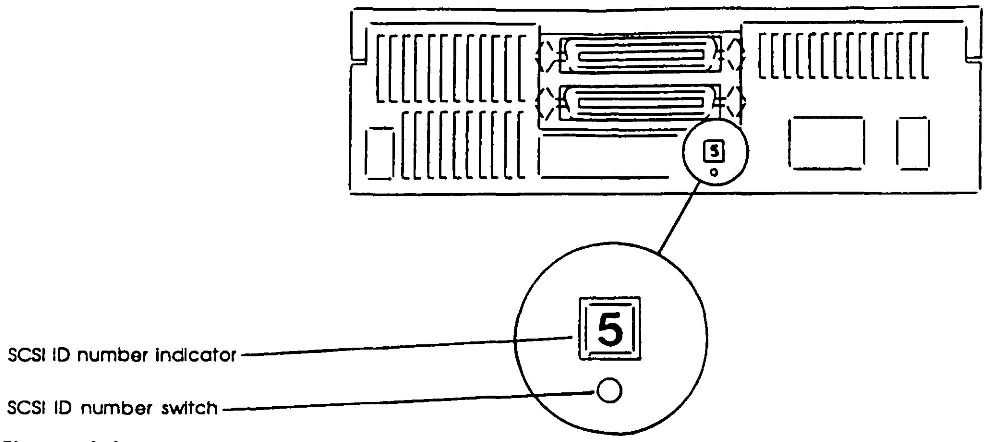


Figure 4-2
SCSI ID number indicator and switch

5. If you are now adding the very first external Hard Disk SC to your system, skip this step and proceed instead to step 6.

However, if you already have one or more external Hard Disk SCs connected to your Macintosh II, follow these steps and refer to Figure 4-3 to add another hard disk.

- □ Disconnect the 50-pin connector of the System Cable from the SCSI connector on an existing external Hard Disk SC, and attach a 50-pin connector of the Peripheral Interface Cable in its place. Press the diamond-shaped wire clips toward the Peripheral Interface Cable connector, and then snap them into the clip brackets to secure the connection.
- □ Attach the other end of the Peripheral Interface Cable to either SCSI connector on the new Hard Disk SC. Snap the clips into the clip brackets.
- □ Attach the free end of the System Cable to the available connector on the new Hard Disk SC. Snap the clips into the clip brackets.
- □ Make sure that the Cable Terminator is connected to the Hard Disk SC at the end of the SCSI chain.

  Note that the end of the SCSI chain is not necessarily the disk with the highest SCSI ID number. The Macintosh II with an internal hard disk is at the beginning of the SCSI chain; the Hard Disk SC at the opposite end of all cabled devices is considered to be the end of the SCSI chain.

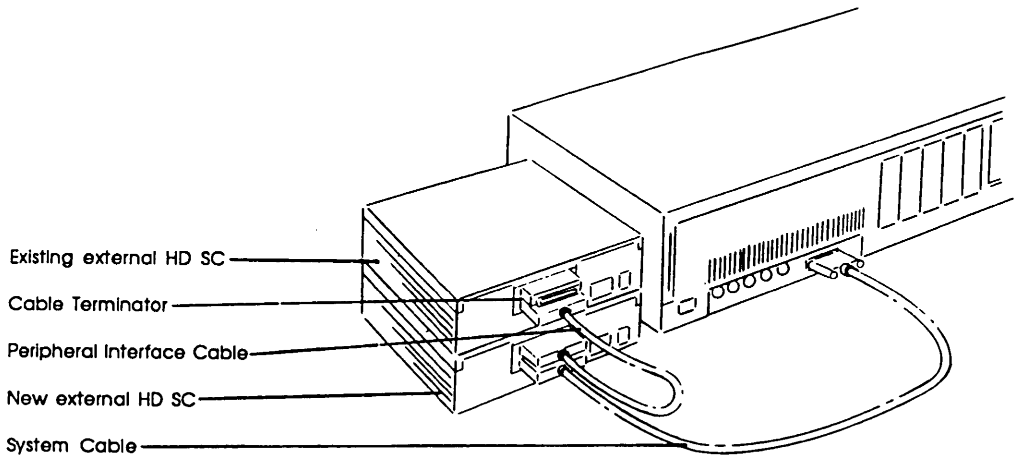


Figure 4-3
Attaching two external Hard Disk SCs

Warning

A Cable Terminator must always be connected to the peripheral device at the physical end of the SCSI chain.

- ☐ If your system has an internal hard disk installed, skip both the remainder of this step and step 6, and proceed now to step 7.

However, if your system has A/UX on an external Hard Disk SC and has no internal hard disk at all, you must ensure that both your new Hard Disk SC and your Hard Disk SC that contains A/UX have Cable Terminators attached. This setup is necessary because the SCSI system requires terminating resistors on the first and last devices in the SCSI chain. Apple internal hard disks have terminating resistors built-in; Apple external Hard Disk SCs do not. Instead, you must use Cable Terminators as terminating resistors for external Hard Disk SCs.

Therefore, to attach a second external Hard Disk SC when there is no internal hard disk installed, perform the following additional steps.

- ☐ Disconnect the System Cable from its SCSI connector on the Hard Disk SC.
- ☐ Attach a Cable Terminator to the 50-pin connector of the System Cable, as shown in Figure 4-4. Snap the clips into the clip brackets to secure the connection.

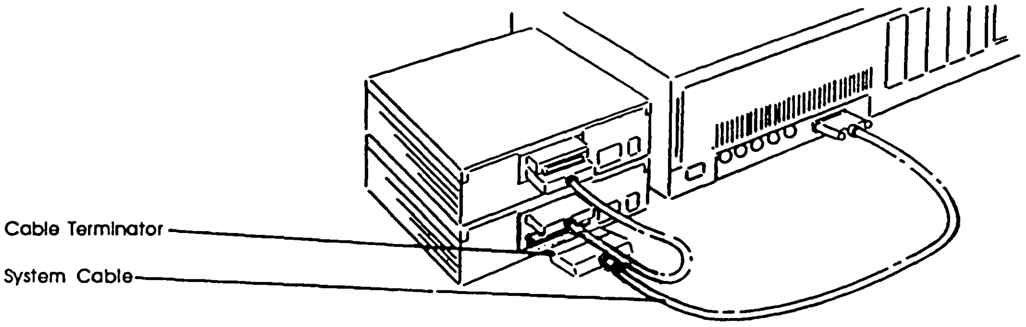


Figure 4-4
Attaching the Cable Terminator to the System Cable
(only when there is no internal hard disk)

- ☐ Attach the Cable Terminator, which is already attached to the System Cable, to the available SCSI connector on the Hard Disk SC. Snap the clips into place to secure the connection. Both of your external Hard Disk SCs should now have Cable Terminators attached, and a Peripheral Interface Cable should connect the disks.
- ☐ For future reference, use a pencil or a pen to draw an arrow that redirects you from the beginning of step 5 to the steps given next. If you add more external Hard Disk SCs or other SCSI devices to your system, you should skip the previous instructions in step 5 and follow those given next instead.

If you attach more external SCSI devices in the future, keep your current Hard Disk SCs as the first and last devices in the SCSI chain, as they already have Cable Terminators attached. Replace all of step 5 with the following instructions.

- □ Disconnect the Peripheral Interface Cable from the SCSI connector on the Hard Disk SC at either end of the SCSI chain. Note that the ends of the SCSI chain are not the necessarily the disks with the highest or lowest SCSI ID numbers; rather, they are the disks at the physical ends of all connected SCSI devices, and therefore ought to have Cable Terminators already attached.
- □ Reattach the free end of the Peripheral Interface Cable to either of the available SCSI connectors on your new SCSI device.
- □ Attach one end of a new Peripheral Interface Cable to the last available SCSI connector on your new SCSI device.
- □ Finally, attach the free end of the new Peripheral Interface Cable to the available SCSI connector on the Hard Disk SC at the physical end of the SCSI chain.

To continue your Hard Disk SC setup, skip step 6 and proceed to step 7 instead.

6. **If you are attaching your very first external Hard Disk SC to the system, follow these instructions.**

   If you already have one or more external Hard Disk SCs attached, you should have followed step 5 above; in which case, you should now skip this step and proceed instead to step 7.

   - □ Attach the 25-pin connector of the SCSI System Cable to the SCSI connector on the back of your Macintosh II, as shown in Figure 4-5. When connecting the System Cable, press the diamond-shaped wire clips toward the connector, and then snap them into the clip brackets to secure the connection.

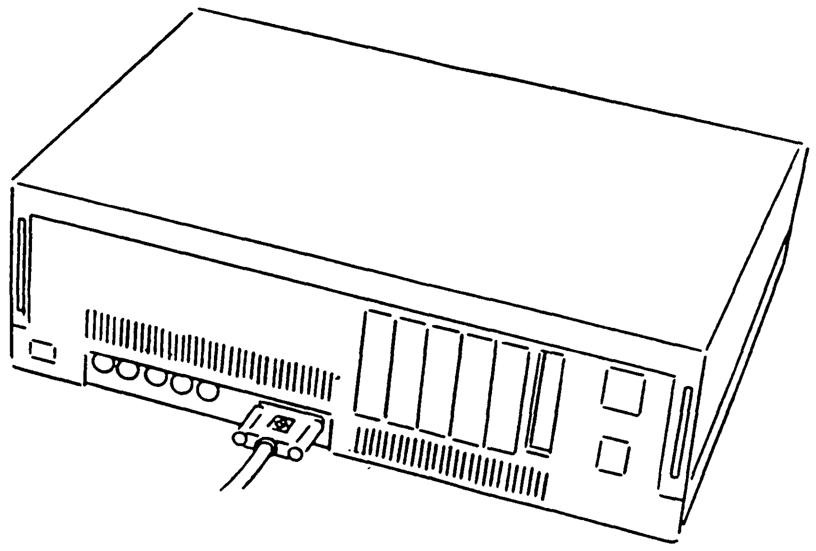


Figure 4-5
Attaching the System Cable to the Macintosh II SCSI connector

Warning

Attach only the SCSI System Cable to the SCSI connector on the computer. To identify the connector, look for the SCSI icon, shown in Figure 4-6, on the rear panel of the computer.

Putting another type of cable, such as an RS-232 connector, in the SCSI connector can seriously damage the SCSI chip inside your computer or the computer itself.

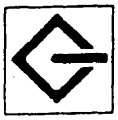


Figure 4-6
SCSI connector icon

- □ Tighten the thumbscrews on the DB-25 connector.
- □ As shown in Figure 4-7, attach the 50-pin connector of the SCSI System Cable to either of the SCSI connectors on the back of the external Hard Disk SC.

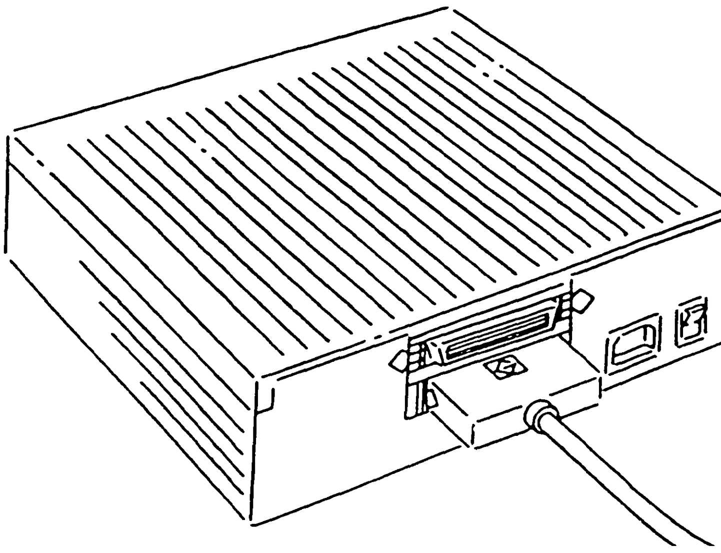


Figure 4-7
Attaching the SCSI System Cable to the Hard Disk SC

- □ Attach the Cable Terminator to the available SCSI connector on the external Hard Disk SC, as shown in Figure 4-8. Snap the clips into the clip brackets to secure the connection.

Warning

A Cable Terminator must always be connected to the peripheral device at the physical end of the SCSI chain.

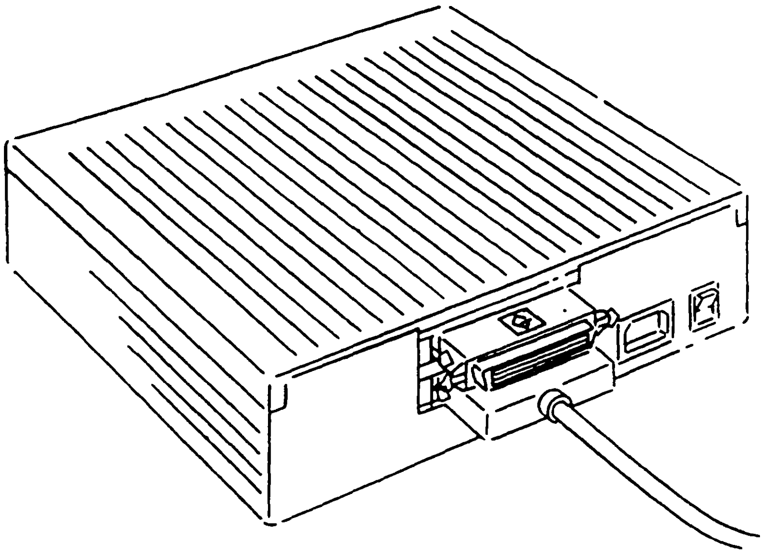


Figure 4-8
Ending the SCSI chain with a Cable Terminator

7. **With all peripheral devices attached to your computer system and *switched off*, attach the power cord to the Hard Disk SC and plug it into a grounded (three-prong) AC outlet.**

   The power switch for all Hard Disk SCs is located to the right of the power receptacle on the back of the device. Press the bottom of the switch to make sure the device is off.

Warning

This equipment is intended to be electrically grounded.

Apple computers and peripheral devices are equipped with three-wire grounding plugs—plugs that have a third (grounding) pin. These plugs will fit only grounding-type AC outlets. This is a safety feature.

If you are unable to insert the plugs into the outlets, contact a licensed electrician to replace the outlets with properly grounded outlets.

Do not defeat the purpose of the grounding plug!

8. **Arrange your components, keeping your external Hard Disk SC away from the front left side of the Macintosh II main unit, and allow adequate ventilation around all components.**

   Keep external Hard Disk SCs away from the front left side of the computer where they risk magnetic interference from the Macintosh II's power supply.

   To ensure proper ventilation while the Hard Disk SC is on, be sure not to block the air vents at the sides and the back of the unit.

Warning

Protect your system from overheating by keeping at least 4 inches clear around all sides of your external Hard Disk SCs and your computer's main unit. Always keep Hard Disk SCs and the main unit flat. Standing them on edge or not allowing room for air to circulate will defeat their cooling design and may eventually damage them.

---

9. **Switch on the external Hard Disk SCs with Cable Terminators attached.**

   You need to switch on the peripheral devices that have terminating resistors attached (the first and last peripheral devices in the SCSI chain) to allow information to pass through the cable system. Begin by turning on the Hard Disk SC with the Cable Terminator attached.

   When you turn on your Macintosh II in a few minutes, the internal hard disk, which has a terminating resistor built in, is automatically turned on. However, if your system does not have an internal hard disk, turn on the other external Hard Disk SC that has an attached Cable Terminator.

10. **If you have additional SCSI devices attached, switch them on now.**

    If you aren't using a device in the middle of the SCSI chain, you can leave it switched off.

Using an external Hard Disk SC for the Macintosh OS

If you are adding an external Hard Disk SC that already contains Macintosh files and you want to access them through the Macintosh OS, skip the next section, "Initializing Your Hard Disk SC," and proceed instead to the section "Using a Hard Disk SC With the Macintosh OS." Otherwise, initializing your disk will erase all of its current Macintosh data.

If you're adding a new disk that does not contain any data, you must follow the instructions in the section "Initializing Your Hard Disk SC," regardless of whether you want it to store Macintosh OS or A/UX files.

---

11. **If your disk is new, or if you used it for other operating systems and now you want it to contain an A/UX file system, follow the steps in the next section to initialize your Hard Disk SC.**

Initializing your Hard Disk SC

To use the Hard Disk SC, you must **initialize** the disk—that is, prepare it to receive information by organizing its surface into tracks and sectors. This section describes how to use the Apple HD SC Setup program to initialize your new disk.

If you have a disk that you've already used with the Macintosh OS or with an Apple II, and if you want to use it for A/UX now, you should initialize it as described in this section. However, back up the data on that disk before continuing with these instructions. It is important to back up your data because reinitializing the disk erases all of its current information. To back up your disk, use a utility like the HDBackup application, which is found on the *Utilities* disk that was shipped with your Macintosh II computer.

1. **Insert the *Hard Disk SC Setup* disk into either of the 3.5-inch internal disk drives on your Macintosh II.**

   The 3.5-inch *Hard Disk SC Setup* disk was shipped with your Hard Disk SC.

   Every Macintosh II has at least one 3.5-inch internal disk drive; your system may have two.

2. **Turn on your Macintosh II.**

   Press the Power On key on the keyboard. The 3.5-inch internal disk drives have startup priority over SCSI disk drives. Therefore, because the *Hard Disk SC Setup* disk is in the internal disk drive and because it contains Macintosh startup files, the Macintosh II starts up from this disk instead of from the A/UX disk.

3. **Double-click the Hard Disk SC Setup icon to open it.**

   Use the mouse to move the pointer to the disk icon labeled *Hard Disk SC Setup* on the right side of your screen as shown in Figure 4-9. Double-click on the icon by quickly pressing and releasing the mouse button twice. On your screen a window opens that shows a System Folder, a Utilities Folder, and the Apple HD SC Setup application icon.

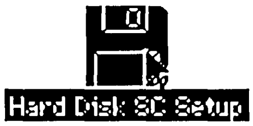


Figure 4-9
Double-clicking Hard Disk SC Setup

4. **Double-click the icon of the Apple HD SC Setup application.**

   Figure 4-10 shows the icon. After you open the application, a dialog box appears, as shown in Figure 4-11.

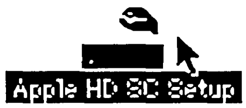
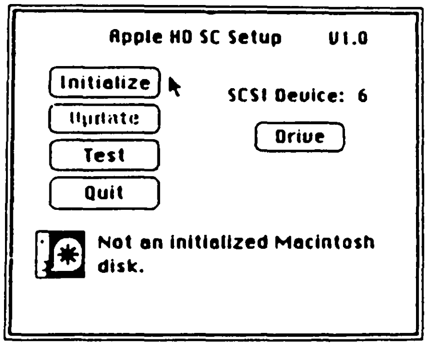


Figure 4-10
Double-clicking Apple HD Setup


Apple HD SC Setup V1.0

SCSI Device: 6

Not an initialized Macintosh disk.

Figure 4-11
The Apple HD SC Setup dialog box

Next to the words "SCSI Device" in the dialog box is a number from 0 to 6. If you have only one external Hard Disk SC, the preset priority number of the drive automatically appears as the SCSI device number.

If you have two or more external SCSI devices connected, make sure that this number matches the SCSI ID number of the Hard Disk SC you want to initialize. If the numbers do not match, use the mouse to move the pointer to the Drive button on the screen and clicking (that is, pressing and releasing the mouse button) until the SCSI numbers do match.

Warning

Make sure that you have chosen the Hard Disk SC you want to initialize, and not some other hard disk. Initializing destroys all information on the hard disk.

---

5. Click Initialize, as shown in Figure 4-12.

Figure 4-12
Clicking Initialize

After you click the Initialize button, another dialog box appears, reminding you that initializing erases all the information from the Hard Disk SC. To continue, click OK as shown in Figure 4-13.

Figure 4-13
Clicking OK

Messages appear in the Setup window, explaining the course of the initialization. Initialization takes several minutes. The length of time depends on the size of your Hard Disk SC.

If you see a message that the Hard Disk SC failed to initialize properly, try again. If you are still unable to initialize, there may be a hardware problem. See your authorized Apple dealer.

6. Name the disk.

A message tells you when initializing has been successfully completed and asks you to name the disk. This name identifies the disk under the Macintosh OS. However, A/UX won't use the name you give here.

You can click OK and leave the disk untitled if you plan to use the disk for storing an A/UX file system. However, if your site has multiple Hard Disk SCs, you may find it helpful to give the disk a name, especially if you share Hard Disk SCs among several Macintosh computers.

You can type up to 27 characters, and you can use any keyboard character except the colon (:). However, the disk name should not begin with a period. Press the RETURN key when you are sure you have entered the right name.

7. **Click Quit, as shown in Figure 4-14.**

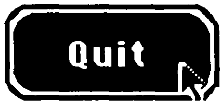


Figure 4-14
Clicking Quit

Clicking Quit returns you to the Finder™ Desktop, where the Hard Disk SC icon appears with the name you gave it.

8. **Choose Restart from the Special menu.**

Use the mouse to move the pointer to Special in the menu bar at the top of the screen. Press the mouse button so that Special is highlighted and a menu of commands appears. Hold the mouse button down and drag the mouse downward until the command Restart is highlighted. Then release the mouse button.

This process causes the Macintosh II first to eject all 3.5-inch disks from their drives, and then to search for the Startup Device. This device will normally be the disk containing A/UX, in which case the sash application is launched and begins counting down the seconds before booting A/UX.

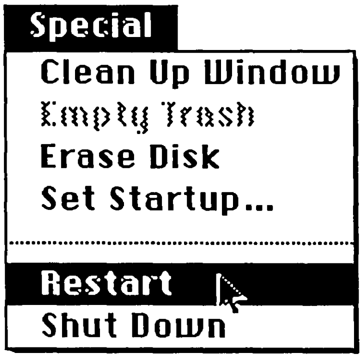


Figure 4-15
Choosing Restart from the Special menu

(If you used the Control Panel desk accessory to change the Startup Device, start sash now by double-clicking its icon.)

9. **If you want to use this disk for Macintosh OS files, interrupt the automatic boot procedure and then read the next section, "Using a Hard Disk SC With the Macintosh OS."**

   Cancel the automatic boot by performing either of the following:

   □ clicking the Cancel button on the screen

   □ holding down the Command key (marked with an Apple icon and a cloverleaf icon) and type a period (.)

10. **If you want to use this disk for an A/UX file system, let A/UX boot automatically, skip the next section, and proceed to the section following titled "Partitioning a Hard Disk SC for an A/UX File System."**

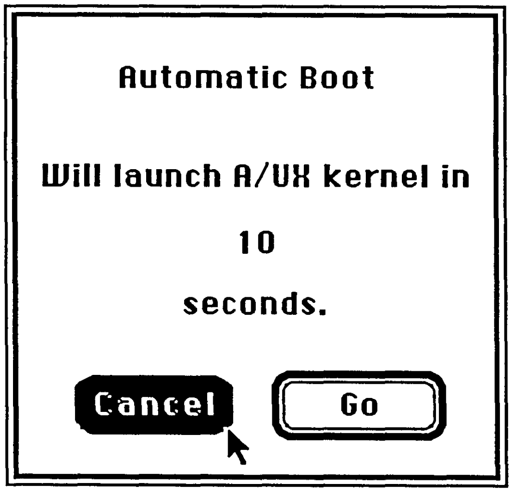
Using a Hard Disk SC with the Macintosh OS

This section is intended to help you use your newly attached external Hard Disk SC for Macintosh Operating System access.

If you connected a Hard Disk SC that already contains Macintosh OS files, make sure that the Hard Disk SC is turned on, and then press the Power On key on the Macintosh II keyboard. When the A/UX automatic boot screen appears, cancel the A/UX boot by performing either of the following:

□ clicking the Cancel button on the screen, as in Figure 4-16

□ holding down the Command key (marked with an Apple icon and a cloverleaf icon) and typing a period (.)


Automatic Boot

Will launch A/UX kernel in

10

seconds.

Figure 4-16
Clicking Cancel

If you just finished initializing your Hard Disk SC, you should have already interrupted the A/UX automatic boot as described earlier.

You should find the `sash#` prompt on your screen. The rest of this section provides instructions for switching between the two operating systems—A/UX and the Macintosh OS—on your computer.

However, if you didn't interrupt the A/UX automatic boot, let A/UX boot to single-user mode; refer to the instructions in "Returning to the Macintosh OS From A/UX" later in this section for assistance in bringing down A/UX and restarting your computer under the Macintosh Operating System.

Deciding which disk to use as the Startup Device

Since you're dedicating this additional Hard Disk SC for the Macintosh Operating System, you need to consider the order in which the SCSI devices are accessed when the Macintosh II starts up. The Macintosh II automatically looks for Macintosh OS startup files in the following places and in the following order:

1. Internal 3.5-inch disk drive number 0
2. Internal 3.5-inch disk drive number 1
3. A Macintosh Hard Disk 20 (a serial, not a SCSI, device)
4. The SCSI hard disk with the highest SCSI ID number, or the hard disk set as the Startup Device with the Control Panel desk accessory
5. Another hard disk with the next-highest SCSI ID number

The A/UX Stand-Alone Shell (`sash`) is a Macintosh application in a Macintosh partition on the factory-loaded A/UX Hard Disk SC. This disk is initially configured to be the Startup Device. Therefore, unless you change the Startup Device with the Control Panel desk accessory, the Hard Disk SC that contains A/UX is always started first. In this case, the `sash` application—being the startup application on this disk—automatically tries to boot A/UX whenever you turn on or restart the Macintosh II.

Alternatively, if you want the Macintosh OS instead of A/UX to come up automatically when you turn on or restart your Macintosh II, change the Startup Device to be the new disk containing the Macintosh OS by performing the following.

□ Make sure there is Macintosh system software (contained in a System Folder) on the Hard Disk SC dedicated to the Macintosh OS. Refer to "Installing System Software" in Appendix A of your Macintosh II owner's guide for information about installing and updating system software on your extra Hard Disk SC.

- ☐ Use the Control Panel desk accessory to set the Startup Device to be the Hard Disk SC dedicated to the Macintosh OS. Refer to "Using the Control Panel" in Chapter 3 of your Macintosh II owner's guide for instructions. Remember that the Hard Disk SC containing A/UX is called `sash partition`; the Hard Disk SC dedicated to the Macintosh OS has the name you gave it when you initialized the disk.

Restarting A/UX from the Macintosh OS

Whenever you are ready to restart A/UX, quit and save all of the work you've been doing under the Macintosh OS.

If `sash` is the startup application on your system and if `sash` is configured to boot A/UX from the correct disk, you can automatically boot A/UX by using the mouse to select Restart—either from the Special menu under the Macintosh Finder or from the Execute menu under the `sash` application.

In any case, you can always boot A/UX by performing the following set of procedures.

- ☐ If the disk `sash partition` is not already open, open it by double-clicking it with the mouse as shown in Figure 4-17.

Figure 4-17
Double-clicking sash partition

- ☐ If the `sash` application is not already running, start it by double-clicking its icon, shown in Figure 4-18. The `sash` application counts down and then boots A/UX to single-user mode.

Figure 4-18
The sash icon

- ☐ If you cancel or interrupt the automatic boot procedure, you can start A/UX by entering the command `boot` at the `sash#` prompt or by using the mouse to select Boot from the Execute menu. Then follow the A/UX startup procedures described in either the *A/UX Installation Guide* or the *A/UX Local System Administration* to bring the system up to multiuser mode.

Returning to the Macintosh OS from A/UX

To access the files under the Macintosh Operating System, you must bring down A/UX and restart the system. Unless you configured the disk containing the Macintosh OS as the Startup Device, you must then interrupt the A/UX automatic boot and quit the `sash` application.

Shutting down A/UX from single-user mode

If your system is already booted and running A/UX in single-user mode, bring A/UX down by

- □ entering the `killall` command to stop all active processes
- □ entering the `sync` command at least twice to flush the system buffers
- □ entering the `reboot` command
- □ interrupting the automatic boot either by clicking the Cancel button on the screen, or by holding down the Command key (marked with an Apple icon and a cloverleaf icon) and typing a period (.)

Note that if you configured the disk containing the Macintosh OS as your startup device, you will automatically start up under the Macintosh OS and you won't need to cancel the A/UX automatic boot as just described.

Shutting down A/UX from multiuser mode

If your system is already booted and running A/UX in multiuser mode, bring A/UX down by

- □ entering the `shutdown` command to return to single-user mode
- □ entering the `sync` command at least twice to flush the system buffers
- □ entering the `reboot` command
- □ interrupting the automatic boot either by clicking the Cancel button on the screen, or by holding down the Command key (marked with an Apple icon and a cloverleaf icon) and typing a period (.)

Note that if you configured the disk containing the Macintosh OS as your startup device, you will automatically start up under the Macintosh OS and you won't need to cancel the A/UX automatic boot.

Quitting `sash`

When you cancel the A/UX automatic boot procedure, you will be in the `sash` application. To exit to the Macintosh Finder, quit `sash` by selecting Quit from the File menu as explained here.

- ☐ Use the mouse to move the pointer to File in the menu bar at the top of the screen.
- ☐ Press the mouse button so that File is highlighted and a menu of commands appears.
- ☐ Hold the mouse button down and drag the mouse downward until the command Quit is highlighted.
- ☐ Release the mouse button.

Instead of using the mouse, you can also quit `sash` by holding down the Command key (marked with an Apple icon and a cloverleaf icon) and typing *q*.

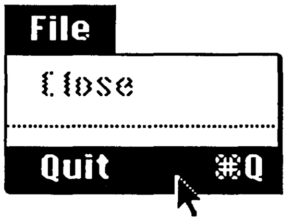


Figure 4-19
Quitting `sash`

You then see the Finder Desktop under the Macintosh Operating System. Icons for your attached disks appear on the right side of the screen. To begin using your new disk, open it by double-clicking its icon. (Refer to your Hard Disk SC owner's guide or your Macintosh II owner's guide for information about setting up files and folders on this disk.)

Since you're using the disk for the Macintosh OS, you can skip the rest of this document, which describes how to configure the Hard Disk SC and A/UX for accessing A/UX file systems from an external Hard Disk SC.

Partitioning a Hard Disk SC for an A/UX file system

A hard disk can be divided into logical sections, called partitions, to accommodate multiple file systems and even multiple operating systems. For example, the A/UX distribution disk contains one partition for the Macintosh OS, one for the A/UX root file system, one for A/UX process paging, and two for storing redundant copies of important files so that Autorecovery can help restore a corrupted system to a good state. However, creating multiple partitions on a disk is a complex and difficult procedure.

Creating a disk partition consists of setting the partition's attributes and establishing a mapping from the partition's logical address to the disk's physical address. You use the dp utility to change or create a disk partition for A/UX.

To avoid the complications of creating multiple partitions on a single disk, this section describes the simplest case: partitioning the entire disk for a single A/UX file system. This configuration should serve the needs of the vast majority of users. However, if you wish to investigate more elaborate partitioning schemes, read this section and then read dp(1M) and pname(1M) in *A/UX System Administrator's Reference*. However, consider yourself warned: creating multiple partitions might require more work than they're worth.

A/UX device names for Hard Disk SCs

The rest of this documentation makes extensive reference to the A/UX device names used by the Hard Disk SCs. A/UX names Hard Disk SCs according to the following convention:

/dev/dsk/c*n*d*m*s*y*

The directory /dev lists all of the devices under A/UX. The subdirectory dsk lists all of the block devices; that is, those like the Hard Disk SCs from which you can mount A/UX file systems.

The value of *n* after c is the SCSI ID of the Hard Disk SC; the value of *m* after d is the number of the sub-drive at that SCSI ID; and *y* after s refers to the slice, or partition, on that disk.

Some controller boards support multiple disk drives off one SCSI ID, and *m* selects that second drive. However, all of the Hard Disk SCs that you connect to the Macintosh II through its built-in SCSI port have separate SCSI ID numbers; none will be identified as sub-drives. For example, the internal hard disk on a Macintosh II is identified by the device name /dev/dsk/c0d0s*y*; the first external Hard Disk SC—when given SCSI ID number 5, as is the convention—is identified as /dev/dsk/c5d0s*y*.

The slice number, *y*, identifies a partition on the disk. Slice 0 always refers to the root partition, and slice 31 always refers to the entire disk.

On Hard Disk SCs that you add to the system, slice 1 is normally used for the swap partition (for temporarily storing portions of processes awaiting execution), and slice 2 is normally used for a usr partition (typically for storing a user file system.)

❖ *Note:* If you use dp to create partitions on slices other than 0, 1, or 2, you must use pname(1M) to associate these partitions with A/UX device nodes.

Using dp to remap the Hard Disk SC partition

A Macintosh-type partition was created on your disk when you initialized it with the Apple HD Setup utility. In this section, you'll use dp to remap that partition as an A/UX file system.

1. **Boot A/UX to multiuser mode.**

   Let the automatic boot procedure of the `sash` application bring A/UX up to single-user mode. (If A/UX does not come up in single-user mode, refer to the next page for assistance.) Then enter the command `init 2` to bring A/UX up to multiuser mode.

   (The entire startup procedure is described in detail in Chapter 2 of *A/UX Local System Administration* and Chapter 6 of the *A/UX Installation Guide*.)

2. **Log in as `root`.**

   You'll need superuser privilege to use several of the following commands.

3. **Use the dp utility to examine the current partition map and partition attributes.**

   □ Replacing *n* with the SCSI ID number of the new Hard Disk SC, enter the following command:

   ```
   dp /dev/dsk/cnd0s31
   ```

   The dp utility returns a message informing you of the partition created on that disk when you ran the Macintosh Hard Disk SC Setup program. 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" 1 partitions, 1 allocated [unknown size]
   Command?
   ```

   ❖ *Note:* If you get a different response, enter an uppercase *Q* to quit dp. Reenter the dp command just given, making sure you specify the correct SCSI ID and that you type the command correctly.

   When using dp, you can quit at any time without making any changes by entering uppercase *Q* until you return to the `root` user prompt. If *Q* doesn't quit, you can also send the *interrupt* character (which, by default, you enter under A/UX by holding down CONTROL while typing *c*) to quit dp without making any changes. You can also enter a question mark (?) at the `Command?` prompt to see a list of available dp commands.

If A/UX doesn't boot to single-user mode

If you get the Finder Desktop under the Macintosh OS or the icon of a blinking question mark instead of the sash application, check the following and correct as necessary.

- □ The disk containing A/UX must be powered on and properly cabled to the Macintosh II. (The disk icon for sash partition will be visible on the right side of the screen if the disk is properly connected.)
- □ If you want A/UX to start up automatically, the disk containing A/UX should be made the Startup Device. Use the Control Panel desk accessory to configure the disk sash partition to be the Startup Device. (This procedure is explained in detail in "Using the Control Panel" in Chapter 3 of your Macintosh II owner's guide.)
- □ If you want A/UX to start up automatically, sash should be made the startup application on the Startup Device. Use the mouse to move the pointer to the sash application and click once. Doing this highlights the sash icon. Then choose Set Startup from the Special menu. This selects sash as the startup application.
- □ If you do not want A/UX to start up automatically, or if it was not previously configured to start up automatically, double-click the sash application. This starts the A/UX automatic boot procedure.

If you receive an error message like

```
Disk c0d0s0 Error: Cannot select SCSI device
generic disk c0d0s0 Fatal Error: Logical block 0
chroot failed
sash#
```

then sash is configured to boot off the wrong Hard Disk SC. From the sash screen, choose General from the Preferences menu, and change the first number in the RootDirectory box to be the SCSI ID number of the hard disk containing A/UX (as explained in Appendix B of the *A/UX Installation Guide*).

- □ In response to the Command? prompt, enter the uppercase letter

  ```
  P
  ```

  to print to your screen information about all allocated partitions on the disk. The dp utility returns output similar to that shown next.

  ```
  Name: "Macintosh", Type: "Apple_HFS"
  Physical: 156353 @ 16, Logical: 156353 @ 0
  Status:
          valid   alloc   in_use  boot
          read    write
  No Block Zero Block
  ```

Particularly notice the number of physical blocks reported in the second line. The number 156353 is the approximate number of blocks you should find on a Hard Disk 80SC; you should find approximately 84277 physical blocks on a Hard Disk 40SC, and approximately 38965 physical blocks on a Hard Disk 20SC.

- ☐ Write down the number of physical blocks reported to you by `dp`. You will use this number as an argument to the `mkfs` command later when you make a UNIX file system on the disk.

4. Use the `dp` utility to change the file type to a UNIX file system.

- ☐ The `dp` utility should again display its `Command?` prompt. Enter a lowercase `c` and the number `0` as shown here:

  ```
  c0
  ```

  This command tells `dp` to change partition 0, which is the only one on the disk. You are now prompted for the data partition map entry (DPME) field:

  ```
  DPME Field?
  ```

- ☐ Enter

  ```
  t
  ```

  to change the file system type. The `dp` utility then prompts whether to partition it as a default UNIX type:

  ```
  Default type?
  ```

- ☐ Answer yes to this prompt by entering the lowercase letter

  ```
  y
  ```

  You are again prompted for the DPME field:

  ```
  DPME Field?
  ```

- ☐ Enter

  ```
  b
  ```

  to establish the partition's block zero block (BZB). You are prompted to provide the cluster number for the BZB:

  ```
  Cluster # [0]:
  ```

- ☐ Enter the number

  ```
  0
  ```

  for the cluster number. You must enter this number; pressing RETURN doesn't send 0 as the default answer. You are now prompted to choose between three file system types:

  ```
  FS type (1=UNIX,2=EFS,3=SFS) [?]:
  ```

- ☐ Enter the number

  `1`

  to specify the file system type as UNIX. You are now prompted to decide whether or not to create the partition as a `root` file system, in which case it will be created on slice 0 (specified as the number following the `s` in the `/dev/dsk/c`*n*`d0s0` device name).

  ```
  Root file system?
  ```

  In response to this prompt, you can answer `y` to create the file system on slice 0; you can answer `n` to create it as a `usr` file system on slice 2 (at the next prompt); or you can answer `n` to this prompt and the next prompt, in which case you must use the `pname(1M)` utility to name and associate the partition with a device node. (Refer to *A/UX System Administrator's Reference* for information about `pname`.)

- ☐ The simplest and easiest way to create and maintain the file system is to make it a `root` file system; to do so, respond to the `Root file system?` prompt with the lowercase letter

  `y`

  You are prompted to decide whether or not to create the partition as a `usr` file system, in which case it will be created on slice 2:

  ```
  Usr file system?
  ```

- ☐ Since you've already created the partition as a `root` file system on slice 0, enter

  `n`

  You are again prompted for the DPME field:

  ```
  DPME Field?
  ```

- ☐ Enter

  `n`

  to begin changing the partition's name. Now you're prompted for a name for the partition:

  ```
  Name [Macintosh]:
  ```

- ☐ Give the partition any useful name, such as

  `A/UX_Partition`

  Again you're prompted for the DPME field:

  ```
  DPME Field?
  ```

- ☐ Enter

  `q`

  to quit the file type configuration module of dp. Doing this returns you to the dp utility's `Command?` prompt:

  ```
  Command?
  ```

5. Examine the new attributes of the partition.

Respond to the `Command?` prompt by entering the uppercase letter

```
P
```

You will see the changes made to the Hard Disk SC's partition. The following is a sample of the kind of output to expect. The number of physical and logical blocks reported in the second line of this output may differ, depending on the size of the Hard Disk SC you partitioned and on the number of available good blocks.

```
DPM Index: 0
  Name: "A/UX Partition", Type: "Apple_UNIX_SVR2"
  Physical: 156353 @ 16, Logical: 156353 @ 0
  Status:
          valid   alloc   in_use  boot
          read    write
  Regular UNIX File System (1)
  Cluster:   0     Type: RFS       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
```

6. Write the changes to the partition and quit `dp`.

In response to the `Command?` prompt that appears on your console, enter the lowercase letters

```
wq
```

to save your partition changes and quit the dp utility. The `root` user's command prompt should reappear on your console.

Making and mounting an A/UX file system

Now that you've initialized your external Hard Disk SC and partitioned it for A/UX, you're ready to make a file system on it and mount it onto your A/UX directory hierarchy.

1. Use `mkfs` to make a file system on the new Hard Disk SC.

The `mkfs` utility constructs a file system on a block device such as a Hard Disk SC. The format of the `mkfs` command you will enter is

`mkfs    /dev/dsk/c`*n*`d0s`*y*   *number_of_blocks*

The variables in this command for which you must supply values are as follows:

- *n,* the SCSI ID number of the disk. This number is normally 5 for the first external Hard Disk SC, or 6 for the second.
- *y,* the disk slice on which you created the A/UX partition. This number is 0 if you created a `root` partition, or 2 if you created a `usr` partition.
- *number_of_blocks,* the number of physical blocks partitioned on the disk. You should have written this number down when you performed step 3 in "Partitioning a Hard Disk SC for an A/UX File System."

Replace these variables with the appropriate values for your system, and enter them into the `mkfs` command. For example, if the SCSI ID number for the disk you are adding is 6, if you created a root partition at slice 0, and if the dp utility reported that your disk has 156353 physical blocks on it, you would enter the following command:

```
mkfs  /dev/dsk/c6d0s0  156353
```

The following message appears:

```
Mkfs:/dev/dsk/c6d0s0?
(DEL if wrong)
```

This gives you 10 seconds in which you can press the DELETE key and cancel the `mkfs` command. Don't press DELETE; within 10 seconds, `mkfs` begins constructing a file system on the partition of your external Hard Disk SC. When `mkfs` is finished, it reports the total number of logical blocks and inodes created on the disk and returns your command prompt.

2. Run `fsck` on the new Hard Disk SC.

To ensure the consistency of the file system you just made, run the `fsck` utility by entering the following command, but be sure to substitute the Hard Disk SC's SCSI ID number for *n* and to substitute the partition's slice number for *y:*

```
fsck  -y  /dev/dsk/cnd0sy
```

For example, if your new external Hard Disk SC has SCSI ID number 6 and the slice number for the partition is 0, enter the command

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

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

3. Make a mount point for the new file system.

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, but any files in the directory will be obscured for the duration of the mount. For example, the empty directory `/mnt` was shipped on your A/UX distribution disk as a convenient mount point. However, you may want to reserve `/mnt` for temporary file systems; for example, you may wish to use `/mnt` for mounting file systems on removable 3.5-inch disks.

The following instructions describe how to name a mount point that refers explicitly to the Hard Disk SC that contains the mounted file system. If you add more than one Hard Disk SC, this process provides a convenient way to identify the physical locations of your file systems.

Replacing the SCSI ID number of the Hard Disk SC for *n,* enter the following command:

```
mkdir /hdscn
```

For example, if your new external Hard Disk SC has SCSI ID number 6, enter

```
mkdir /hdsc6
```

This directory is the mount point where you will attach the file system contained on your new Hard Disk SC.

4. Mount the new file system.

Replacing the SCSI ID number of the Hard Disk SC for *n,* and replacing the partition slice for *y,* enter the following command:

```
mount -v /dev/dsk/cnd0sy /hdscn
```

The command returns a message informing you that the file system on your additional Hard Disk SC is now mounted as the `/hdsc`*n* directory.

For example, if your new external Hard Disk SC has SCSI ID 6 and the slice number for the partition is 0, enter

```
mount -v /dev/dsk/c6d0s0 /hdsc6
```

5. Create a `lost+found` directory for the new file system.

□ Change directory to the new file system's mount point. Replacing the SCSI ID number of the Hard Disk SC for *n,* enter the command

```
cd  /hdscn
```

For example, if your new external Hard Disk SC has SCSI ID 6, enter

```
cd  /hdsc6
```

- ☐ Type the following command to create a `lost+found` directory on the new file system:

  ```
  mklost+found
  ```

  This command reports its progress as it creates a `lost+found` directory in the top directory of the new file system, and creates and removes empty files to make slots available for future use.

  In the future, when you run `fsck` on this file system after a system crash, and there are files that have become unconnected in its directory structure, `fsck` will place the files in this `lost+found` directory. Files connected to `lost+found` are named by their inode number instead of their original filenames. You can use the `ls -l` command to determine who the files belong to, and then tell the owners to copy the files back to their directories under their original filenames.

6. Test the file system.

To verify that your new file system is accessible, try writing to it and reading from it with the following commands. (Substitute the SCSI ID number of your new Hard Disk SC for *n*.)

```
cp /etc/passwd  /hdscn/mount.test
cat  /hdscn/mount.test
```

Your system's password file should scroll across your screen. Enter the command

```
rm /hdscn/mount.test
```

to remove the test file.

If the password file did not appear, reenter the commands in this step and make sure you 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.

7. Put the file system entry in `/etc/fstab`.

Begin by making a backup copy of `/etc/fstab`.

```
cp /etc/fstab  /etc/fstab.old
```

When changing an important system file like this, it is always a good idea to back it up so that if something goes wrong, you can always copy the backup to its original name and restore your system to its previous state.

To add a new entry to `/etc/fstab`, type

```
cat >> /etc/fstab
```

and press RETURN. The cursor will move to a new line, but you will not see a prompt. Replacing *n* with the SCSI ID number of the additional Hard Disk SC, and replacing *y* with the partition slice, type the following line (but don't press RETURN yet):

```
/dev/dsk/cnd0sy  /hdscn  5.2  rw  1  2
```

For example, if your new external Hard Disk SC has SCSI ID 6 and the partition is located at slice 0, type

```
/dev/dsk/c6d0s0  /hdsc6  5.2  rw  1  2
```

Before you press RETURN, make sure you have entered the line correctly. If you mistyped, press DELETE to back over and erase previous characters, and then retype the correct ones.

After you've double-checked that you typed the correct information, press RETURN. The cursor moves to a new line, but you will not see a prompt.

Hold down CONTROL and type *d*.

The command prompt will reappear. The line you typed earlier is now included in the file `/etc/fstab`, and the file system on your additional Hard Disk SC will be mounted automatically whenever you boot A/UX to multiuser mode.

The next time you reboot A/UX to multiuser mode or bring it down to single-user mode and then back to multiuser mode, enter the command

```
mount
```

This command, with no options or arguments, reports all of the mounted file systems. The device name for your new Hard Disk SC and its mount point should appear in this brief report.

❖ *Note:* If the `mount` command does not return a listing for your new Hard Disk SC, then its entry in `/etc/fstab` is incorrect. You may have already noticed that something is amiss. (For example, `fsck` may have returned a SCSI error message when checking the file system.) Return `/etc/fstab` to its previous state with the command

```
cp  /etc/fstab.old  /etc/fstab
```

and then repeat all of step 7.

Addendum: How A/UX boots from hard disks

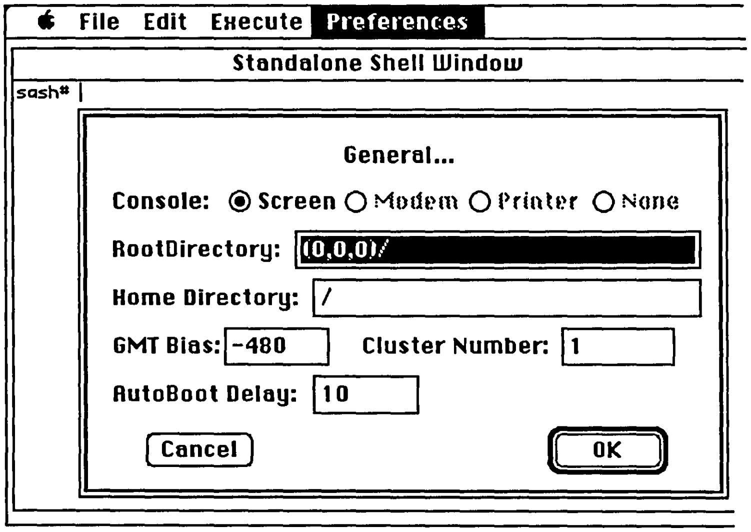
A/UX has been designed to be booted off of any SCSI disk, from SCSI ID 0 through SCSI ID 6. The A/UX kernel accepts boot device information from `sash`. This design is unlike many other UNIX kernels, which frequently have the boot device "hardwired" into the kernel and require use of `adb` or similar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

The A/UX kernel requires two pieces of information about the disk that it will be running off of: the `root` file system and the swap area. The A/UX kernel does not know about disks in general. Rather, it knows that it can call one of several block device drivers (by using the major number to select one of the block device drivers), and that it can pass to that block device driver a parameter (the minor number) indicating to the block device driver just where to go.

The block device driver (in the case of the A/UX initial release, the SCSI disk driver) uses the minor number to select the SCSI ID number and a slice—that is, a partition—of the disk. The minor number is meaningless to the A/UX kernel; it merely passes it along to the block device driver.

The steps of the algorithm that A/UX uses for booting are as follows:

1. The `sash` application (a Macintosh OS 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. You can change these major and minor numbers by pulling down the Preferences menu and choosing General from `sash`. The General dialog box shown in Figure 4-20 appears.



Console: [✓ Screen] Modem Printer None

RootDirectory: (0,0,0)/

Home Directory: /

GMT Bias: -480 Cluster Number: 1

AutoBoot Delay: 10

Figure 4-20
The General dialog box under `sash`

The RootDirectory box (see Figure 4-21) is used to change the information that is passed to the A/UX kernel. In the initial release of A/UX, only the first two numbers of the set (0,0,0) are used. The last number is currently ignored.

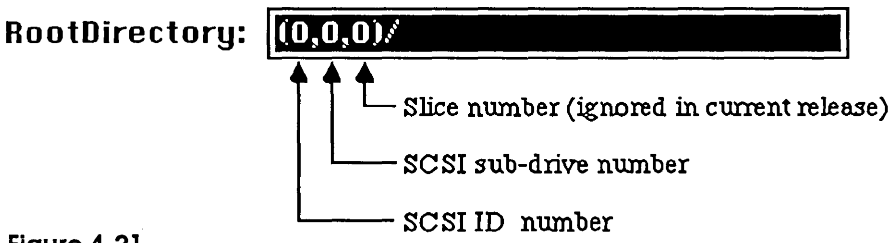


Figure 4-21
The RootDirectory box

2. When the A/UX kernel begins executing, it examines the internal variables `rootdev`, `swapdev`, and `pipedev`. These variables each contain a major and a minor number, and each variable identifies (to the block device driver) the `root` file system, the swap area, and the pipe file system. (The pipe file system is almost always the same as the `root` file system.)

   If any of these variables contain an "illegal" value (0xff), then A/UX uses the values left by `sash` in a specified area of main memory. As stated previously, you can change these values by selecting General from the Preferences menu of `sash`.

   The A/UX kernel as initially configured has the variables `rootdev`, `swapdev`, and `pipedev`, each set to (0xff,0xff). You can use the `kconfig`(1) program to set these variables to a predetermined value; in which case, you can have "hardwired" in the `root` file system or swap areas.

3. The A/UX kernel then calls the device driver, asking for blocks to be read and written. The minor number is passed along to the block device driver, which breaks it down into the SCSI ID number and the slice number. The first time the block device driver is requested to read or write on a slice, it must build the slice information table entry.

   There are 32 active slices possible for a disk drive. By convention, slice 0 is always the `root` file system, slice 1 is always the swap area, slice 2 is a `usr` file system, and slice 31 is always the entire disk.

   The first read or write for slice 0 or slice 1 causes the block device driver to see if the disk has a partition table on it. If it does, the partition table is scanned to look for entries identifying the `root` file system and swap area. If found, the partition entry information is loaded into the slice information table entry.

4. If the disk has no partition entry table, and when the block device driver sees a read or write request the first time for either slice 0 or slice 1, slice 0 and slice 1 are automatically configured to a set number of blocks based upon the total number of blocks on the disk.

5. If no partition entry is found, then the disk is not recognizable to A/UX. You will probably receive a SCSI error message when trying to access the disk.

Section 5

A/UX Toolbox

The A/UX Toolbox is a library that enables a program running under A/UX to call the User Interface Toolbox routines built into the Macintosh II ROMs. The A/UX Toolbox lets you write A/UX programs that take advantage of the standard Macintosh user interface tools, such as the Window Manager, Menu Manager, and Dialog Manager. This library is described in *A/UX Toolbox: Macintosh ROM Interface.*

The A/UX Toolbox also supports the basic elements of the native Macintosh Operating System (OS).

Limitations at the first release (1.0)

At first release, the A/UX Toolbox is not feature-complete. The following limitations exist:

- No support for color displays. The color QuickDraw calls are available, but they function as if the user had the display in 1-bit-per-pixel mode.
- No printing. Calls to `PrOpen()` will fail.
- No shared libraries. All Toolbox applications will have to be relinked for the next release of the Toolbox.
- No layering. Only one Toolbox application can be run at a time.

Resource format and file changes

Since *A/UX Toolbox: Macintosh ROM Interface* was published, the format and file-naming conventions of resource files have been changed. Resource files (with the filename extension `.res`) described in the manual appear in the release either as an AppleSingle file (with the filename prefix %) or as part of an AppleDouble file. See the following description of AppleSingle and AppleDouble file formats and the manual page for `rcnvt(1)` in *A/UX Command Reference.*

Formats for Apple files in foreign file systems

❖ *Note:* This section presents proposed file formats. The information was originally presented in a draft proposal dated October 23, 1987. This information is subject to change.

Apple Computer, Inc., is proposing two standards for representing files on foreign file systems, with the goal of preserving all attributes of the file's home file system on other file systems that do not support the same attributes.

Because of the varying needs of different file systems, a single format seems to be inadequate to cover all cases. However, two closely related formats can serve most needs. Although the primary impetus for developing these formats is storing Macintosh files on file systems that do not support the notion of two forks, the proposed formats are general enough that you can use them to represent a file from any file system on any other file system.

The two formats are called *AppleSingle* and *AppleDouble.*

In AppleSingle format, all contents and attributes of a file are kept in a single file on the foreign file system. For example, both forks of a Macintosh file, the Finder information, any associated icons, and so on are arranged in a single file with a simple structure. This format is intended to be used primarily as a storage format—that is, for cases in which the Macintosh file must be stored on a foreign file system and later reconstructed into a true Macintosh file.

AppleDouble format is more appropriate for applications in which the users of the foreign file system might want to modify the contents of the file. Since most Macintosh applications keep the file data in the data fork, AppleDouble format saves the contents of the data fork in one file. AppleDouble format keeps all other file attributes in a separate file.

Specifically, Apple's proposal does not rule out the possibility of building applications that can access and modify AppleSingle-format files. Nor does it preclude using AppleDouble format for simple storage of Macintosh files. This proposal merely presents them as alternatives.

The only assumption this proposal makes is that each file system on which these file formats will be supported allows the creation of a simple file: an uninterpreted stream of bytes.

AppleSingle and AppleDouble formats are not directly related to the AppleTalk® Filing Protocol (AFP). While one of their possible uses is as the storage format for non-Macintosh–based AFP servers, it was not the primary motivation behind their development.

If you are building an AFP server (or an NFS or other server that supports Macintosh computers), then you may wish to use one of the formats as your internal file storage format. However, the choice is yours. It does not matter what format is used within your application as long as, externally, the files appear as they are supposed to.

There are at least two basic reasons for using either the AppleSingle format or the AppleDouble format:

- As a standard for transferring files between host computers (interhost transfers). For example, Macintosh files could be easily and completely shipped among heterogeneous systems if they all understand one of these common formats. You can use any existing electronic mail system or file transfer utility without modification.
- As a standard for operating on foreign files within a single host (intrahost transfers). In the near future, for example, A/UX applications may build and manipulate Macintosh resource forks (perhaps in a cross-development system). If a set of users of a host computer wishes to write Macintosh-aware applications, the users need to agree on a common storage format, such as the AppleSingle or the AppleDouble format.

The following discussion uses these terms:

- *home file system:* the file system for which the file's contents were created. For example, an A/UX application could create an AppleSingle file that comprises a resource fork and a data fork and that contains a MacWrite-formatted document. The home file system for such a file is the Macintosh file system, because the file is intended to be compatible with a Macintosh application. In most cases, where a file is created and used on the same file system, the home file system is that system.
- *foreign file system:* the other file system that will store or process the file. An AppleSingle or AppleDouble file is usually a representation of a file's contents on the foreign file system.

AppleSingle format

An AppleSingle file contains a header followed by data. The header consists of several fixed fields and a list of entry descriptors, with each descriptor pointing to an entry. Apple defines these standard entries:

- □ Data Fork
- □ Resource Fork
- □ Real Name (name in the home file system)
- □ Comment
- □ Icon
- □ File Info

Each entry is optional and may or may not appear in the file.

An AppleSingle-formatted file follows this structure:

Header	Magic number	4 bytes
	Version number	4 bytes
	Home file system	16 bytes ASCII encoded
	Number of entries	2 bytes
For each entry	Entry ID	4 bytes
	Offset	4 bytes
	Length	4 bytes

The magic number field is modeled after the magic number feature in UNIX. The field is intended to be used in whatever way the foreign file system distinguishes a file as AppleSingle format. The magic number for AppleSingle format is $00051600 or 0x00051600.

The version number field denotes the version of AppleSingle format in case the format evolves (more fields may be added to the header). The version described here is version $00010000 or 0x00010000.

The home file system is a fixed-length, 16-byte ASCII string that is not preceded by a length byte but is possibly padded with blanks.

The home file system header has one of these Apple-defined values:

Macintosh	`'Macintosh'`	or	`$4D616369 $6E746F73 $68202020...`
ProDOS	`'ProDOS'`	or	`$50726F44 $4F532020 $20202020...`
MS-DOS	`'MS-DOS'`	or	`$4D532D44 $4F532020 $20202020...`
UNIX	`'Unix'`	or	`$556E6978 $20202020 $20202020...`
VAX/VMS	`'VAX VMS'`	or	`$56415820 $564D5320 $20202020...`

Apple welcomes suggestions for other file systems that should be included in this list.

The number of entries field tells how many different entries are included in the file. The field contains an unsigned 16-bit number, and may be zero. If the number is nonzero, then that number of entry descriptors immediately follows this field.

For each entry, the entry descriptor indicates what the entry is, where the entry is located in the file, and how big the entry is.

Apple has defined a set of Entry IDs and corresponding values, as follows:

Data Fork	1	Standard Macintosh data fork
Resource Fork	2	Standard Macintosh resource fork
Real Name	3	The file's name in the home file system
Comment	4	Standard Macintosh comment
Icon, B&W	5	Standard Macintosh black-and-white icon
Icon, Color	6	Standard Macintosh color icon
File Info	7	File information, such as attributes
Finder Info	9	Standard Macintosh Finder Info

Apple reserves the range of Entry IDs from 0 to $7FFFFFFF for future use. The rest of the range is available for other systems to define their own entries. Apple will not arbitrate the use of the rest of the range.

Icon entries will probably not appear in most files since they are typically stored as a bundle in the application file's resource fork.

The File Info entry is different for each home file system. For Macintosh HFS, the entry is 16 bytes long and consists of three long integer dates (creation date, last modification date, and last backup date) and a long integer containing 32 Boolean flags. Using the bit-numbering scheme where bit 0 is the least-significant bit and 31 is the most-significant bit, bit 0 of the Macintosh Finder Info entry is the Locked bit; bit 1 is the Protected bit. Formats for other file systems (MS-DOS, UNIX, ProDOS®, and others) are still to be defined.

The actual data representing the entry must be in a single contiguous block. The block is pointed to by the offset field, which is an unsigned 32-bit number indicating the byte offset from the start of the file to the start of the entry. The entry length is also an unsigned 32-bit number representing the length in bytes. The length may be zero.

After some number of entry descriptors, the actual entry data appears. The entries could appear in any order, but because the data fork is the entry that is most commonly extended, Apple strongly recommends that the data fork entry always be kept last in the file to facilitate its extension.

Apple also recommends that those entries that most often need to be read, such as Finder Info, Real Name, and Dates, be kept as close as possible to the header to maximize the probability that a read of the first one or two blocks of the file will retrieve these entries.

It is possible to have holes (unused space between entries) in the file. To find where the holes are, you must take the list of entry descriptors and sort them into increasing offset order. If the offset field of an entry is greater than the offset plus the length of the previous entry, then a hole exists between the entries. You can make use of such holes; for example, if a file's comment is 10 bytes long, you could create a hole of 190 bytes after the comment field to allow easily for the comment to expand to its maximum length of 200 bytes. Because an AppleSingle file may contain holes, you must find each entry by getting its offset from its entry descriptor, and not by assuming that it begins after the previous entry.

Byte ordering in the file header fields will follow 68000 and 68020 conventions.

AppleDouble format

AppleDouble format is the same as AppleSingle format, except that the data fork is kept in a separate foreign file. The file containing the data fork is called the *AppleDouble data file,* and the other file is called the *AppleDouble header file.*

The AppleDouble data file consists of only the standard Macintosh data fork, with no extra header at all. The AppleDouble header file has exactly the same format as the AppleSingle file, except that it does not contain a data fork entry. The magic number of an AppleDouble header file differs from that of an AppleSingle file so an application can tell whether or not it needs to look elsewhere for the data fork. The magic number for AppleDouble format is $00051607 or 0x00051607.

The entries in the header file could appear in any order, but since the resource fork in this case is the entry that is most commonly extended, Apple strongly recommends that the resource fork entry always be kept last in the file. The data fork is easily extended, because it resides by itself in the AppleDouble data file.

If it is possible on the foreign file system, you could create a new type of entry that pointed to the AppleDouble data file to make the file easy to find.

Filename conventions

AppleSingle format specifically does not include an algorithm for generating the AppleSingle filename from the file's real name. The foreign file systems of interest differ quite a bit in filename syntax, and the file's real name can be kept as an entry within the AppleSingle file.

The same is generally true for AppleDouble data filenames. However, Apple is proposing a standard for deriving the AppleDouble data file and AppleDouble header filenames from the file's real name. Because filename syntax differs in the various file systems, the proposed standard varies according to file system.

UNIX

To generate the AppleDouble data filename, use character substitution to replace any illegal characters with an underscore (_) and, if necessary, truncate the filename to 14 characters.

To generate the AppleDouble header filename, prefix a single percent sign (%) to the AppleDouble data filename. If necessary, truncate the last character to keep the filename within the legal length range.

ProDOS

To generate the AppleDouble data filename, use character substitution or deletion to remove illegal characters, and use truncation if necessary to reduce the length of the name to two characters less than the maximum filename length.

To generate the AppleDouble header filename, add the two characters uppercase-R and period (R.) as a prefix to the AppleDouble data filename.

MS-DOS

To generate the AppleDouble data filename, use character substitution or deletion to remove illegal characters, and use truncation if necessary to reduce the length of the name to eight characters. Then add the MS-DOS extension that is most appropriate to the file (for example, '.TXT' for a pure text file).

To generate the AppleDouble header filename, add the extension '.ADF' (for AppleDouble file) to the eight-character filename.

AppleDouble name derivations will be defined for all other file systems of interest. These name derivations will allow applications running on the foreign file system (and human users as well) to see easily which files are AppleDouble pairs. Users who know the derivation could rename or move the files in order to preserve the connection between the two. However, there is no guaranteed way to prevent one file of the pair from being inconsistently renamed, moved, or deleted.

Section 6

A/UX Kernel Messages

This section describes two types of messages generated by the A/UX kernel:

- panic messages
- warning messages

A/UX kernel panic messages

An A/UX kernel panic can result from any one of four basic causes:

- Kernel resources. A kernel resource panic, such as `kmem_alloc` (not enough free memory for kernel alloc), is caused when some preset resource limit is exceeded. Most often, these limits are imposed either by the physical hardware or by kernel parameters set at boot time. You can overcome some hardware limitations by expanding your system (physically adding more memory, for example). You can also modify some kernel parameters (see `kconfig`(1M) in *A/UX System Administrator's Reference* and the section "Changing Kernel Parameters" in these release notes).
- Catastrophic failure. A catastrophic failure panic is normally caused by a hardware failure (such as memory errors), which can interfere with normal execution of instructions or can destroy kernel data. The panic thus provides software validity checks on the kernel environment. In the case of such an error, the system is halted before additional damage can occur or can be transferred to the root file system.

  As with any major system error, large data loss is possible. The best protection is frequent backups. While major malfunctions are rare, spending a few minutes a day can save months of work in case of catastrophic failure. You should never see a catastrophic kernel failure. If you do, contact your Apple representative.

- Introduced error. Errors are ordinarily introduced to the kernel only during program development, usually by the addition of a new device driver. The panic supplies useful diagnostic feedback to the programmer. An everyday user should never see these panic messages.

  Sometimes, though, a coding error in the kernel shows up only under unusual circumstances. These are the "bugs" in the kernel. If you see a kernel panic caused by an introduced error during normal operation, report it to your Apple representative.

- Corrupt file system. In some cases, it is possible to destroy portions of the root file system through the use of broken hardware, bad drivers, or even unreasonable raw device IO requests. (The best protection against this kind of damage is to run `fsck`(1M) when booting the system.) If one of these panics occurs, run `fsck` and repair the file system as necessary. Damage to the file system must be fairly severe to produce these panics. Be prepared to resort to the re-creation of the file system and complete restoration from your backups (see *A/UX Local System Administration* and the manual page for `fsck`(1M) in *A/UX System Administrator's Reference*).

Table 6-1 lists all of the possible kernel panic messages, describes the most likely cause of each, and offers suggestions for dealing with those problems that you can address.

Table 6-1
A/UX kernel panic messages

Message	Type	Comments
`"CPU can't page fault properly"`	Catastrophic failure	Check hardware configuration.
`"Error in standalone driver mkbad implementation"`	Catastrophic failure	
`"Error in standalone driver onestate() implementation"`	Catastrophic failure	
`"SCSI manager software error in state table"`	Catastrophic failure	
`"accept"`	Catastrophic failure	
`"ae6int"`	Catastrophic failure	Unexpected interrupt from an "empty" NuBus slot. Look for the problem on NuBus.

Table 6-1 (continued)
A/UX kernel panic messages

Message	Type	Comments
`"alloc:bad size"`	Catastrophic failure	Corrupt file system. Requested file system block not standard size. Check drive; run `fsck`(1M) to rebuild as necessary.
`"allocbuf"`	Kernel resources	Argument size greater than available buffer size. Increase variable `SBUFSIZE` with `kconfig`(1M).
`"bad clist count"`	Catastrophic failure	
`"bad major number in gdrestart"`	Introduced error	
`"blkdev"`	Catastrophic failure	
`"bread"`	Catastrophic failure	Check hard disk and cables.
`"bread:size 0"`	Catastrophic failure	
`"breada"`	Catastrophic failure	Check hard disk and cables.
`"breadrabp"`	Catastrophic failure	Check hard disk and cables.
`"brealloc"`	Catastrophic failure	
`"bwrite"`	Catastrophic failure	
`"call of function at` *`location`*`"`	Introduced error	Printed message routine when a pointer to a function is not initialized properly; for example, `(*func)() == 0`.
`"cannot allocate buffer cache"`	Kernel resources	Add physical memory and increase variable `NCLIST` with `kconfig`(1M) or reduce memory use.
`"cannot allocate buffer headers"`	Kernel resources	Add physical memory and increase variable `NBUF` with `kconfig`(1M) or reduce memory use.
`"cannot allocate character buffers"`	Kernel resources	Add physical memory and increase variable `NCLIST` or `NBUF` with `kconfig`(1M) or reduce memory use.
`"cannot allocate page hash table"`	Kernel resources	Add physical memory or reduce memory use.
`"clget:null client"`	Catastrophic failure	
`"closef"`	Catastrophic failure	
`"devtovp_badop"`	Catastrophic failure	

Table 6-1 (continued)
A/UX kernel panic messages

Message	Type	Comments
`"direnter:target directory link count"`	Corrupt file system	Corrupted file system. Run `fsck`(1M) as necessary to correct.
`"dirmakeinode:no attributes"`	Catastrophic failure	
`"dirprepareentry: invalid slot status"`	Catastrophic failure	
`"dirprepareentry:new block"`	Catastrophic failure	
`"dnlc_pruge:zero vp"`	Catastrophic failure	
`"dup biodone"`	Catastrophic failure	
`"error in standalone driver implementation"`	Catastrophic failure	Check drive being used at time of error.
`"findreg -- no match"`	Catastrophic failure	
`"free:bad size"`	Corrupt file system	Requested file system block is not standard. Check drive, run `fsck`(1M) as necessary to correct.
`"freeproc -- cannot find child on chain"`	Catastrophic failure	
`"fstat"`	Corrupt file system or also Catastrophic failure	Unknown file type in status read. May also be a catastrophic failure. Check drive. Run `fsck`(1M) as necessary to correct.
`"getfreehdr"`	Kernel resources	Unable to find a free block header. Increase the variable `NPBUF` with `kconfig`(1M).
`"getmajor"`	Catastrophic failure	
`"getmp:bad magic"`	Catastrophic failure	
`"getpages -- pbremove"`	Catastrophic failure	
`"IO error in swap"`	Catastrophic failure	Check hard disk and cables.
`"icmp len"`	Catastrophic failure	
`"icmp_error"`	Catastrophic failure	
`"iget:bad dev"`	Catastrophic failure	
`"iget:bad fs"`	Catastrophic failure	

Table 6-1 (continued)
A/UX kernel panic messages

Message	Type	Comments
`"in_control"`	Catastrophic failure	
`"interrupt stack overflow"`	Introduced error	Interrupt stack is located in the UDOT area. Even though you can increase the size of the UDOT (and stack), this is not recommended. Instead, reduce use of the stack in the driver interrupt service routine being worked on.
`"ip_init"`	Catastrophic failure	
`"iput"`	Catastrophic failure	
`"irele"`	Catastrophic failure	
`"itrunc1"`	Catastrophic failure	
`"itrunc2"`	Catastrophic failure	
`"iunlock"`	Catastrophic failure	
`"kernel memory management error"`	Catastrophic failure	Generic 68000 bus error message. Kernel bus error has occurred. Check all hardware.
`"kmem_alloc"`	Kernel resources	Not enough free memory for kernel allocation request; more kernel activity than allowed by current memory size. Add physical memory and adjust the variable MAXCORE with `kconfig`(1M).
`"kmem_free"`	Catastrophic failure	
`"kmem_free: block already free"`	Catastrophic failure	
`"kmem_free block already free as neighbor"`	Catastrophic failure	
`"kmem_free_intr"`	Catastrophic failure	
`"m_clalloc"`	Catastrophic failure	
`"m_clalloc MPG_SPACE"`	Catastrophic failure	
`"m_copy"`	Catastrophic failure	
`"m_cpytoc"`	Catastrophic failure	
`"m_more"`	Catastrophic failure	

Table 6-1 (continued)
A/UX kernel panic messages

Message	**Type**	**Comments**
`"main.c -- copyout of icode failed"`	Kernel resources	Add physical memory or reduce memory use.
`"main.c -- iicode growreg failure"`	Kernel resources	Add physical memory or reduce memory use.
`"mbinit"`	Catastrophic failure	
`"mcldup"`	Catastrophic failure	
`"mclput"`	Catastrophic failure	
`"missing phys()"`	Catastrophic failure	
`"mminit"`	Catastrophic failure	
`"nfs_badop"`	Catastrophic failure	
`"no procs"`	Catastrophic failure	
`"pinsert dup"`	Catastrophic failure	
`"piusrreq"`	Catastrophic failure	
`"psig"`	Catastrophic failure	
`"psig action"`	Catastrophic failure	
`"ptcwrite"`	Catastrophic failure	
`"raw_usrreq"`	Catastrophic failure	
`"realvtop:invalid search depth"`	Catastrophic failure	
`"region count list overflow"`	Catastrophic failure	
`"remrq"`	Catastrophic failure	
`"revarp:no mbufs"`	Kernel resources	Increase the variable `NMBUFS` with `kconfig`(1M).
`"rfs_lookup"`	Catastrophic failure	
`"rootfsmount: cannot mount root"`	Catastrophic failure	Check hard disk and cables.
`"rootmount: cannot find root vnode"`	Catastrophic failure	Check physical memory. Check hard disk and controller.
`"rtfree"`	Catastrophic failure	

Table 6-1 (continued)
A/UX kernel panic messages

Message	Type	Comments
`"rwip"`	Catastrophic failure	
`"rwvp:zero size"`	Catastrophic failure	
`"sbappendaddr"`	Catastrophic failure	
`"sbdrop"`	Catastrophic failure	
`"sbflush"`	Catastrophic failure	
`"sbflush 2"`	Catastrophic failure	
`"soaccept:NOFDREF"`	Catastrophic failure	
`"soclose:NOFDREF"`	Catastrophic failure	
`"socreceive flush PR_ATOMIC"`	Catastrophic failure	
`"socreceive get MT_RIGHTS"`	Catastrophic failure	
`"socreceive no packets"`	Catastrophic failure	
`"sofree dq"`	Catastrophic failure	
`"software error in SCSI finish reset"`	Catastrophic failure	
`"software error unknown command in mkbad()"`	Catastrophic failure	
`"soisconnected"`	Catastrophic failure	
`"soreceive no aname"`	Catastrophic failure	
`"soreceive no rights or address"`	Catastrophic failure	
`"sorecieve no mbufs"`	Kernel resources	Increase the value of the variable `NMBUFS` with `kconfig`(1M).
`"sosend"`	Catastrophic failure	
`"sptalloc -- ptmemall failed"`	Catastrophic failure	
`"sptfill -- ptmemall failed"`	Catastrophic failure	

Table 6-1 (continued)
A/UX kernel panic messages

Message	**Type**	**Comments**
"sptseggrow:hard limit exceeded"	Introduced error	Too many sptalloc requests; reduce them.
"startup -- swapadd failed"	Catastrophic failure	Swap device failed. Check hard disk and cables.
"svfs_badop"	Catastrophic failure	
"svfs_inactive"	Catastrophic failure	
"svfs_ioctl"	Catastrophic failure	
"svfs_readdir"	Catastrophic failure	
"svfs_select"	Catastrophic failure	
"svfs_statfs"	Corrupt file system	Possible bad file system. Check drive for valid file system; run fsck(1M).
"syscall0"	Catastrophic failure	
"syscall1"	Catastrophic failure	
"tcp_output"	Catastrophic failure	
"tcp_output REXMT"	Catastrophic failure	
"tcp_pulloutofband"	Catastrophic failure	
"tcp_usrreq"	Catastrophic failure	
"timeout table overflow"	Catastrophic failure	
"too many softcalls"	Catastrophic failure	
"udp_usrreq"	Catastrophic failure	
"uicp 3"	Catastrophic failure	
"uiomove"	Introduced error	Bad iovector passed to uiomove. Recheck code.
"uipc 1"	Catastrophic failure	
"uipc 2"	Catastrophic failure	
"uipc 4"	Catastrophic failure	
"unexpected kernel trap"	Catastrophic failure	

Table 6-1 (continued)
A/UX kernel panic messages

Message	**Type**	**Comments**
"unimplimented command in scsicmd"	Catastrophic failure	
"uninitialized drive in gdaltinit"	Introduced error	
"unknown command in gdcmd"	Catastrophic failure	
"unknown input in choosetask"	Catastrophic failure	
"unknown input to scsi task"	Catastrophic failure	
"unknown partition state in gdpartinit"	Catastrophic failure	
"unknown state in gdinit_ret"	Catastrophic failure	
"unknown state in gdstart"	Introduced error	
"unknown state in sdreturn"	Catastrophic failure	
"unknown task state in drive init"	Catastrophic failure	
"unknown task state in gdpartinit"	Catastrophic failure	
"unp_connect2"	Catastrophic failure	
"unp_disconnect"	Catastrophic failure	
"unp_externalize"	Catastrophic failure	
"unreconized state in gdaltinit"	Introduced error	
"ureadc"	Catastrophic failure	
"uwritec"	Catastrophic failure	
"vattr_to_nattr"	Catastrophic failure	
"vfault--bad dbd_type"	Catastrophic failure	
"vfs_remove"	Catastrophic failure	

Table 6-1 (continued)
A/UX kernel panic messages

Message	Type	Comments
`"vfs_remove: unmounting root"`	Catastrophic failure	
`"vfs_unlock"`	Catastrophic failure	
`"viaclrius called from non-interrupt"`	Catastrophic failure	
`"video interrupt"`	Catastrophic failure	
`"vn_rel"`	Catastrophic failure	
`"vno_lock"`	Catastrophic failure	
`"vno_unlock:EXLOCK"`	Catastrophic failure	
`"vno_unlock:SHLOCK"`	Catastrophic failure	
`"wakeup p_stat"`	Catastrophic failure	
`"xalloc -- bad magic"`	Catastrophic failure	

A/UX kernel warning messages

An A/UX kernel warning message may be generated by several places in the kernel, including the I/O device drivers and Memory Manager.

The kernel warning messages listed here are not part of a kernel panic, and can provide information or a warning to the user.

Kernel warning messages are classified according to the following types.

- ☐ Kernel status. These messages are usually printed at startup time and provide information on the currently running kernel, such as the kernel version or its configuration. Even though these messages are useful, you can generally ignore them.
- ☐ Fatal error. A serious kernel error has occurred; a kernel panic message is likely to follow.
- ☐ Kernel error. A kernel error has occurred. While additional errors may follow, this error was not fatal to the kernel. A user process may have been killed as a result.
- ☐ I/O status. A device driver has reported on some device status. A user may be interested in this information because it normally gives such information as device number and block size.

- I/O error. A device driver has reported an error condition. Normally, these errors are fatal only to the process making the current request, which may be a request to put 10 MB on a 1-MB floppy disk. However, failure of the root file system disk can bring everything to a dead stop. Look for a kernel panic to follow errors on the root file system disk.
- I/O extended. Some device drivers have additional error messages that are more verbose then the normal IO error messages. These "extended" error messages are used primarily for debugging and are turned on by setting some variable in the driver. The user will not normally see these or need to use them.

Table 6-2
A/UX kernel warning messages

Message	Type	Comments
`"%s - swpuse count overflow"`	Kernel error	Number of processes sharing a page of swap space has exceeded the capacity of the reference counter (currently 256). This error is very unlikely in a demand paged system, such as A/UX.
`"%s on bad dev %o(8)"`	I/O error	Generic kernel driver error message. Currently not used by any A/UX device drivers.
`"%s: Setuid execution not allowed"`	Kernel status	The `setuid` (set user ID) function is not allowed on this NFS file system. This is an optional security feature that may be set according to the exported file system.
`", giving up"`	I/O error	This message appears after the `"NFS server %s not responding"` message to inform the user that an NFS connection has been given up.
`", still trying"`	I/O status	This message appears after the `"NFS server %s not responding"` message to inform the user that an NFS connection is still being attempted.
`"A/UX kernel created %s"`	Kernel status	Kernel status message printed at system startup or reset.
`"Can't allocate message buffer`	Kernel error	Ran out of memory trying to initialize message queues. Reduce kernel memory usage with `kconfig`(1M).

Table 6-2 (continued)
A/UX kernel warning messages

Message	Type	Comments
`"Changing free page high water mark from %d to %d"`	Kernel status	Kernel variable `GETSPGHI` has been reset; use `kconfig`(1M) to reduce it. Unless you have a very good idea of what you are doing, you should not change this variable from its standard value.
`"Changing free page low water mark from %d to %d"`	Kernel status	Kernel variable `GETSPGLOW` has been reset; use `kconfig`(1M) to increase it. Unless you have a very good idea of what you are doing, you should not change this variable from its standard value.
`"DANGER: Out of swap space. Needed %d pages"`	Kernel error	You should never see this message. If it does appear, treat it the same as `"WARNING: Swap space running out. Needed %d pages"`
`"DANGER: mfre map overflow %x: lost %d items at %d"`	Kernel error	Kernel structure has run out of memory. Increase variable `NSPTMAP` with `kconfig`(1M).
`"MC68881 Floating Point Coprocessor ID %d"`	Kernel status	Indicates that the floating-point processor is present (during boot time).
`"NFS %s failed for server %s: %s"`	I/O error	Unable to make connection with remote server. Check remote system; it may be down, or may not have NFS daemons running.
`"NFS server %s not responding"`	I/O status	NFS remote server is not responding. Check the remote server; it may be down or may not have NFS daemons active. Also check hardware.
`"NFS server %s ok"`	I/O error	Connection with NFS remote server remade.
`"NFS write error %d on host %s fs %o file %d"`	I/O error	An error occurred during a write to the NFS remote host system. Check the remote machine, or contact the system administrator for the remote machine.

Table 6-2 (continued)
A/UX kernel warning messages

Message	Type	Comments
`"NFS write error: on host %s remote file system full"`	I/O error	Remove files as needed to make space or contact host's system administrator.
`"Region table overflow"`	Kernel error	Increase the size of the variable `NREGION` with `kconfig`(1M) or add physical memory to the system.
`"Streams space not available"`	Fatal error	There is not enough memory to set up and run streams. Normally, no other error messages will follow because the kernel will be unable to continue to print to the console. Add physical memory to the system or reduce kernel memory use with `kconfig`(1M).
`"System Buffers are more than 90% of remaining memory. UNIX kernel may be unstable -- may need to adjust NBUFS with kconfig"`	Kernel status	Kernel buffers are defined to be so large with `kconfig`(1M) that they take up more than 90% of memory. Increase physical memory or reduce buffer sizes with `kconfig`(1M).
`"Unsupported block size of %d bytes"`	I/O error	The physical block size of a disk is not supported by A/UX. Reformat the device or find another device that can be supported. This error should not occur with standard Apple hardware.
`"WARNING: Swap space running out. Needed %d pages"`	Kernel error	Tried to get more swap space when needed, but was unable to remove unused sticky program from swap. Processes may die if they try to grow. Increase swap space by reconfiguring the hard disk, reduce the number of sticky programs, or reduce the size or number of in-core processes.
`"Warning: No Video Board Found"`	Fatal error	This message suggests that the video board is missing or broken. This message will appear at boot time and only through the first serial (`/dev/tty0`) port.

Table 6-2 (continued)
A/UX kernel warning messages

Message	**Type**	**Comments**
`"ae%d spurious interrupt"`	I/O error	The EtherTalk card generated an interrupt, but its interrupt status register is 0.
`"ae%d transmitter frozen -- resetting"`	I/O error	The EtherTalk card did not receive a transmit complete interrupt within 2 minutes after transmitting a packet.
`"ae%d: ethernet address not found"`	I/O error	The EtherTalk card could not read its address from slot ROM.
`"ae%d: init failed`	I/O error	The EtherTalk card did not respond to a stop command by asserting RST in its interrupt status register.
`"ae%d: can't handle af%d"`	I/O error	The interface was handed a message with addresses formatted in an unsuitable address family. The packet was dropped.
`"ae6_intr: Receive overflow warning"`	I/O error	An overflow condition was noted by the EtherTalk card on incoming packets.
`"ae6int: interrupt from slot %d"`	Fatal error	The EtherTalk card received an interrupt from a slot number that is less than 9 or greater than 14 (legal range for slot interrupts is between 9 and 14).
`"ae6int: Rcv overflow, lost %d packets"`	I/O error	The EtherTalk card missed the stated number of packets because no buffer space was available.
`"authkern_marshal: xdr_authkern failed"`	Kernel status	An `xdr`(3N) routine failed while attempting to deserialize user area credentials.
`"bad auth_len gid %d str %d auth %d"`	Kernel status	An `xdr`(3N) routine failed while attempting to deserialize user area credentials. The numbers given should be the `xdr` length of the serialized group's array, host name, and authentication credentials.
`"bad block %d, ino %d"`	I/O error	Target block resides outside of normal file system boundary. Run `fsck`(1M) to verify file system.

Table 6-2 (continued)
A/UX kernel warning messages

Message	Type	Comments
"bindresvport: couldn't alloc mbuf"	Kernel status	An mbuf could not be allocated to bind to a reserved port for kernel RPC.
"cku_recvfrom: no body!"	I/O error	Tried to check for free memory on nonexistent memory segment.
"clntkudp_create - Fatal header serialization error."	Kernel error	An xdr(3N) routine failed while attempting to serialize the kernel RPC call message header.
"clntkudp_create: socket bind problem"	Kernel error	An attempt to bind a kernel RPC socket to a reserved port has failed.
"clntkudp_create: socket creation problem"	Kernel error	An attempt to allocate a socket for a kernel RPC failed.
"datalock - can't lock %d pages"	Kernel error	Ran out of available memory. Increase physical memory or reduce kernel memory use with kconfig(1M).
"datalock(stack) - %can't lock %d pages"	Kernel error	Ran out of available memory. Increase physical memory or reduce kernel memory use with kconfig(1M).
"device 0x%: bad dir ino %d at offset %d:%s"	I/O error	Kernel found a bad directory entry in the file system. Run fsck(1M) to verify file system.
"drive c%dd%d has inadequate capacity of %d blocks"	I/O error	Drive has reported an unreasonable device size.
"drive c%dd%d is not a disk"	I/O error	The SCSI protocol defined device type, as reported by the device hardware, says that this is not a disk drive. Check the hardware configuration. This error should never occur when using standard Apple hardware.
"duplicate IP address!! sent from Ethernet address: %x:%x"	I/O error	The Ethernet software has received a packet that claims to be from the same IP address as the current machine. Check the file /etc/hosts for conflicts.

Table 6-2 (continued)
A/UX kernel warning messages

Message	Type	Comments
`"exec error: u_error %d pn_path "`	Kernel error	An error occurred while performing an exec on a new process. The kernel was unable to return to the calling process.
`"file: table is full"`	Kernel error	File descriptor table is full. Reduce number of open files or increase the value of the variable `NFILE` with `kconfig`(1M).
`"floppy: %d floppy %s Drive %d has %d head%s floppy: drive %d is %s drive"`	I/O status	Printed at boot time, this message gives information on floppy hardware attached to the system.
`"inode: table is full"`	Kernel error	Incore inode table is full. Reduce number of inodes in use or increase the value of the variable `NINODE` with `kconfig`(1M).
`"ku_fastsend: No source address"`	Kernel error	No internet address could be found for the interface specified in the allocated routine.
`"ku_fastsend: dup ip/udp hdr MGET failed"`	Kernel error	Failed to let memory act as a buffer for the EtherTalk card.
`"ku_fastsend: frag MGET failed"`	Kernel error	Failed to let memory act as a buffer for the EtherTalk card.
`"ku_fastsend: if_output failed: error=%d, am=%x"`	Kernel error	An interface's output routine indicated a failure. The error returned is indicated. Other information is for debugging purposes only.
`"ku_fastsend: ip/udp hdr MGET failed"`	Kernel error	Failed to let memory act as a buffer for the EtherTalk card.
`"ku_recvfrom: len = %d"`	Kernel error	An NFS packet was received with a length greater than `UDPMSGSIZE`.
`"lo%d: can't handle af%d"`	I/O error	The interface was presented with a message using addresses formatted in an unsuitable address family. The packet was dropped. The only address family currently supported is AF_INET (INTERNET).

Table 6-2 (continued)
A/UX kernel warning messages

Message	Type	Comments
`"m_expand returning 0"`	Kernel error	Someone tried to allocate `mbuf` buffers twice. Also caused when the IPC code cannot get additional memory for buffers. Add physical memory or reduce use.
`"memregadd: adding memory"`	Kernel status	Overlooking some physical memory according to the value of the variable `MAXPMEM`.
`"msginit - can't get %d pages"`	Kernel error	Ran out of memory trying to initialize message queues. Reduce kernel memory use with `kconfig`(1M).
`"nfs read: failed, errno %d"`	Kernel status	Read request from NFS server failed. Check local and remote hardware and connections.
`"nfs write: failed, errno %d fh %o %d"`	Kernel status	Write request from NFS server failed. Check local and remote hardware and connections.
`"pcbsetaddr failed %d"`	Kernel error	The number given is the error number returned by the call `in_pcbsetaddr`.
`"proc on q"`	Kernel status	The kernel attempted to enable a process that was already enabled to run.
`"proc: table is full"`	Kernel error	Process table is full. Reduce the number of processes in use or increase the value of the variable `NPROC` with `kconfig`(1M).
`"procdup - can't get %d pages for udot"`	Kernel status	A kernel memory allocation request failed. Add more memory or run fewer processes.
`"puterrno: unmapped UNIX error %d"`	I/O error	An error unknown to this kernel occurred during an NFS operation. Check remote hosts for errors.
`"pvalidate(%x,%x,%x) failed"`	Kernel error	The kernel tried to load a page table entry for an invalid region. You should never see this error. If it does occur, check for hardware problems.

Table 6-2 (continued)
A/UX kernel warning messages

Message	Type	Comments
`"qattach: out of queues"`	Fatal error	There is not enough memory to set up and run streams. Normally, no other error messages will follow because the kernel will be unable to continue. Add physical memory to the system or reduce kernel memory use with `kconfig`(1M).
`"revarp: Requesting Internet address for "` `"Found Internet address %x"`	Kernel status	Status message on finding the ARP address. This message appears only if it took more than two attempts to find the ARP. It is possible for multiple occurrences of the first message to appear, but the second message should appear only once.
`"rwvp: short write. resid %d vp %x bn %d"`	Kernel status	Less data was taken in NFS write than was requested. Check sizes in `/etc/fstab` for NFS remote server.
`"shmctl - couldn't lock %d pages in memory"`	Kernel error	Couldn't get enough memory. Add physical memory or reduce use.
`"sptreserve: No kernel virtual space"`	Kernel error	Kernel ran out of free memory while trying to grow its segment table. The kernel is likely to crash soon after this message appears. Add more physical memory to the system.
`"sptseggrow: no memory"` `"sptseggrow: no memory 2"`	Kernel error	Kernel ran out of free memory while trying to grow its segment table. The kernel is likely to crash soon after this message. Add more physical memory to the system.
`"sptseggrow: soft limit exceded"`	Kernel status	The kernel has allocated an unusually large amount of virtual memory (checked to `SPTCOUNT` defined in `/sys/useg.h`).
`"strioctl: illegal ioctl ack cell"`	I/O error	Unknown stream `ioctl` acknowledge was returned to stream head. Check hardware and communication connections. If using an AST card for AppleTalk, check for error in downloading of software.

Table 6-2 (continued)
A/UX kernel warning messages

Message	Type	Comments
`"stropen: out of streams"`	Fatal error	There is not enough memory to set up and run streams. Normally, no other error messages will follow because the kernel will be unable to continue. Add physical memory to the system or reduce kernel memory use with `kconfig`(1M).
`"stropen:out of queues"`	Fatal error	There is not enough memory to set up and run streams. Normally, no other error messages will follow because the kernel will be unable to continue. Add physical memory to the system or reduce kernel memory use with `kconfig`(1M).
`"svckudp_send: xdr_replymsg failed"`	Kernel error	An `xdr`(3N) routine failed while attempting to serialize a reply to an NFS request.
`"sxt cannot allocate link buffers"`	Kernel error	Kernel is unable to allocate memory. Increase physical memory or reduce the kernel memory use with `kconfig`(1M). This error will most likely be fatal.
`"textlock - can't lock %d pages"`	Kernel error	Ran out of available memory. Increase physical memory or reduce kernel memory use with `kconfig`(1M).
`"total memory size: %d bytes available memory: %d bytes":`	Kernel status	Printed at startup or reset.
`"tp->t_maxfc reduced to %d."`	Kernel status	`MAXSC` was changed on the running kernel, which is not recommended.
`"uinter: process %d tried to delete %d's layer"`	Kernel error	This most likely ends in a panic.
`"uptalloc - can't get %d page"`	Kernel error	Kernel has run out of memory. Reduce kernel memory usage or add physical memory.
`"useracc - couldn't lock page"`	Kernel error	Couldn't lock a page into memory for IO request.

Table 6-2 (continued)
A/UX kernel warning messages

Message	**Type**	**Comments**
`"xdr_mbuf: long crosses mbufs !"`	Kernel error	An `xdr`(3N) routine failed while attempting to serialize or deserialize a block buffer. It found a long integer crossing a block buffer boundary.
`"xdr_mbuf: putlong, long crosses mbufs !"`	Kernel error	An `xdr`(3N) routine failed while attempting to serialize or deserialize a block buffer. It found a long integer crossing a block buffer boundary.
`"can't get mbuf: xdr_rrok failed"`	I/O error	Kernel failed to get an `mbuf` buffer. Reduce use or increase variable `NMBUFS` with `kconfig`(1M).
`"xdrmbuf_putbuf: mclgetx failed"`	Kernel error	Kernel failed to get buffer for a network transfer.

Section 7

Terminal Display Problems

The terminal type of the A/UX Initial Console Emulator is set to the default value of mac2. Your terminal type can be reset automatically to VT100 when you use rlogin, tip, cu, or another data communications utility to log into a non-A/UX system, because the non-A/UX system does not know about the mac2 terminal type.

One immediate effect of this type change is that the scrolling region of the screen is reduced to 23 or 24 lines, which is much smaller than the 35-line Macintosh II region. After you log off the remote system, your terminal type might still be set to VT100. If your terminal type seems wrong after a remote login, check it with this command:

```
echo $TERM
```

If the response is not mac2, reset your terminal type:

- If you are using the Bourne shell (sh) as your login shell, you can reset your terminal type with the command

  ```
  TERM=mac2
  ```

- If you are using the C shell (csh), use the command

  ```
  setenv TERM mac2
  ```

After your terminal type is correctly set to mac2, reset the scrolling region with the command

```
tset
```

If there is obsolete material on your screen, especially material outside of the scrolling region, you can clear the screen with the command

```
clear
```

After you run a graphics package, such as X Windows, under the A/UX Initial Console Emulator, your screen may be left in a strange state. To repaint the screen and restore the console to its normal state, enter the command

```
screenrestore
```

Section 8

Serial Port Difficulties

Your terminal might drop characters while you are simultaneously using either of the serial (`tty`) ports and backing up files to floppy disks or using the Ethernet. The serial ports have lower priority than the floppy disk driver or the network, making it possible to overrun the hardware buffer and lose input characters. This phenomenon is most noticeable at higher baud.

Section 9

RS-232 Cables

This section describes how to make RS-232 cables that will be connected to conventional terminals and computers. A conventional cable is a DB-25P/S (where *P* stands for *plug* and *S* stands for *socket*) configured as either a DTE or a DCE circuit. (For more information, see John E. McNamara, *Technical Aspects of Data Communication,* Second Edition, Digital Press, 1982.) To make this cable, start with the cable (part number M0187) used to connect an Apple ImageWriter® II to a Macintosh Plus, and cut the cable in half. Then wire the cable to a DB-25 connector according to the diagram in Figure 9-1. Depending on which half you start with, the M0187 cable will have one of the two color codes shown on the left side of the figure.

The DCE cable shown in Figure 9-1 has been used successfully at Apple for connection to VT100 terminals, Zenith Z19 terminals, and the DZ11 ports on a DEC UNIBUS communications multiplexor. The cable illustrated in Figure 9-2 is for the less common DTE connector. Note that the cable in Figure 9-2 does not present modem control signals, but that it is wired as a "null modem."

❖ *Note:* Pins 1 and 22 of the DB-25 connector are not used for either cable.

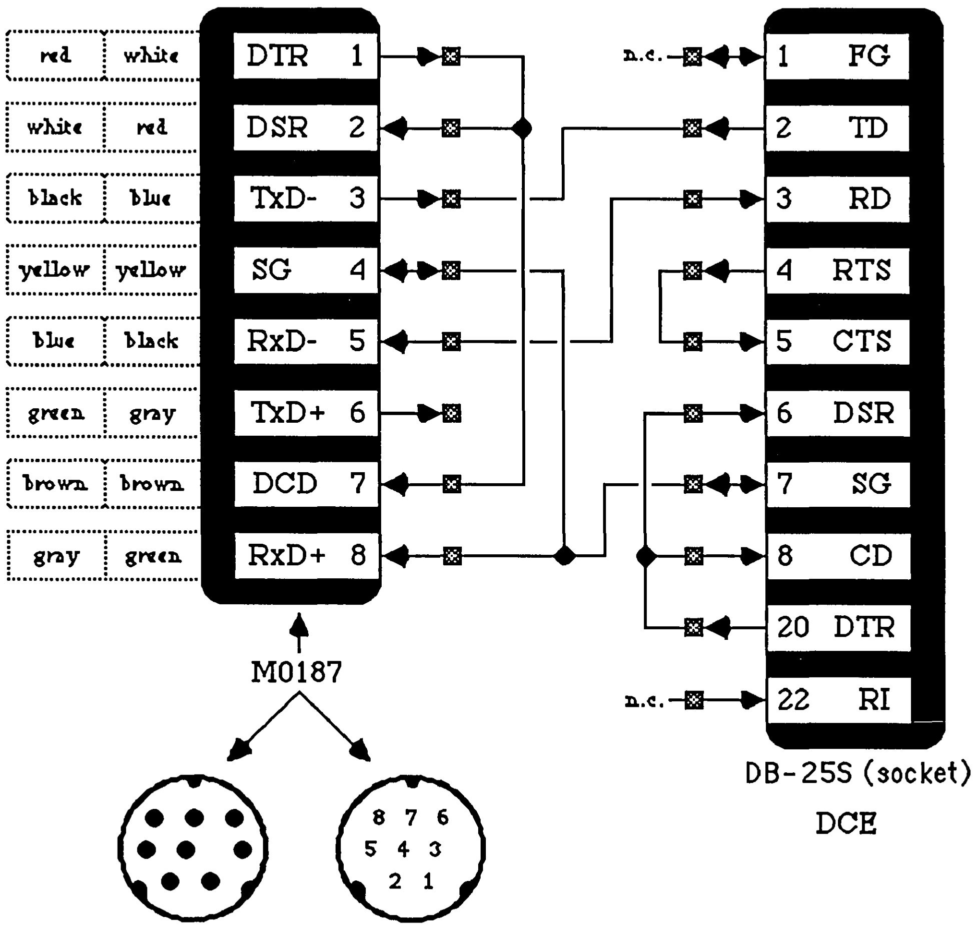


Figure 9-1
RS-232 DCE cable for Macintosh II

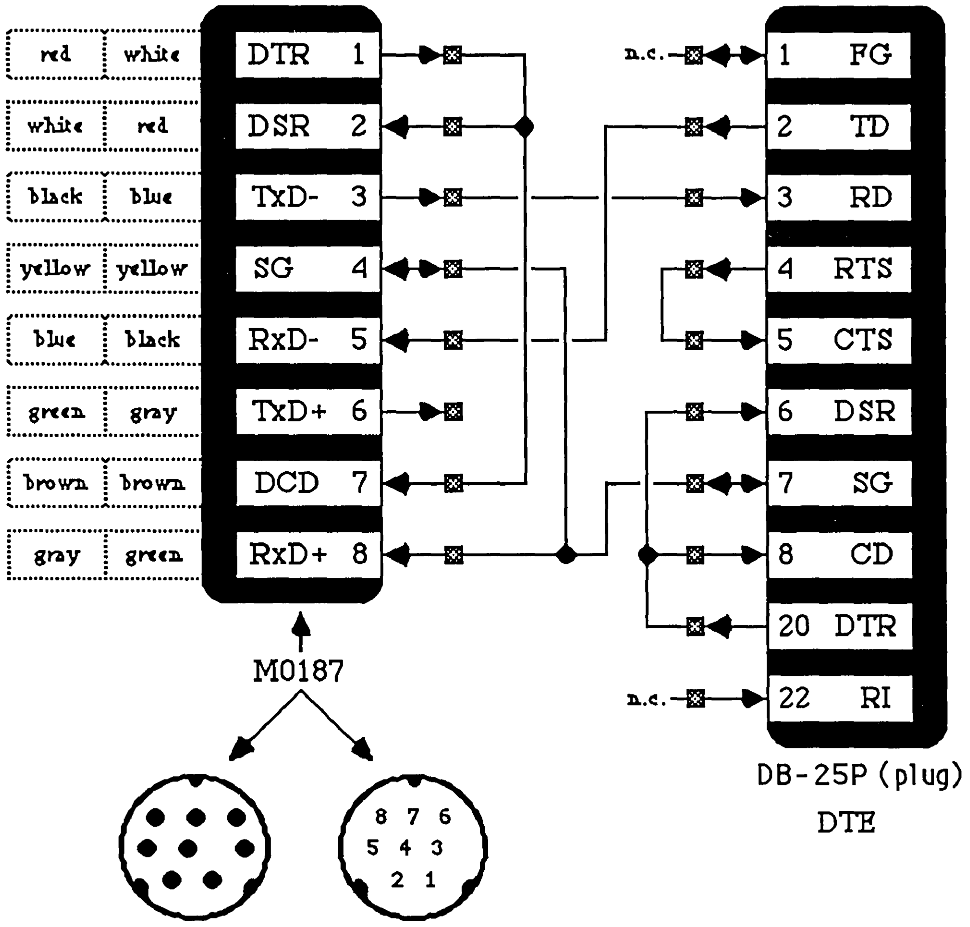


Figure 9-2
RS-232 DTE cable for Macintosh II

Section 10

Manual Pages

This section contains a list of minor corrections to manual pages and a collection of replacement manual pages. These corrections apply to the command reference pages in *A/UX Command Reference, A/UX Programmer's Reference,* and *A/UX System Administrator's Reference.*

Errata

This section lists minor changes to the manual pages in the A/UX reference manuals. Please mark these changes on the affected pages, or mark the pages with a reference to these errata. The on-line manual pages are correct.

`checkmm(1)`

The command `/usr/bin/checkmm1` is invoked by `checkmm(1)`. Add a reference to `checkmm1` under FILES on the `checkmm(1)` page. The `checkmm1` command is not meant to be executed by users.

`domainname(1)`

The man page for `domainname(1)` states that the binary can be found in `/usr/bin`. The correct pathname is `/bin/domainname`.

eqn(1)

The program checkeq(1) is documented on the page for eqn(1). Add a reference to checkeq(1) to the page for eqn(1), and add this line to the page:

checkeq is a related program that reports missing or unbalanced delimiters and .EQ/.EN pairs.

Amend the FILES section to include /bin/checkeq.

ex(1)

The text editor e (in /usr/bin/e) is another name for ex(1). Add to the ex(1) page a reference to e.

finger(1)

Insert the following paragraph above the words *finger options include:*

finger may be used to look up users on a remote machine. To do so, specify *name* as *user*@*host*. If *user* is not supplied, a listing in standard format is provided for the remote machine.

more(1)

The page(1) command is a slight variation on more(1) and should be referenced on the same page. On the page for more(1), add /bin/page to the FILES section, and add to the SYNOPSIS section the line:

page *more_arguments*

To the DESCRIPTION section, add this paragraph:

page functions similarly, except that the screen is cleared before each screenful is displayed (but only if a full screenful is displayed), and $k - 1$ rather than $k - 2$ lines are displayed in each screenful, where k is the number of lines the terminal can display.

put(1)

A/UX does not support this command. Mark this page of the manual for deletion.

take(1)

A/UX does not support this command. Mark this page of the manual for deletion.

uucp(1)

The shell scripts uudemon.day, uudemon.hr, and uudemon.wk are part of the uucp package. Add the following shell scripts to the FILES section on the page for uucp(1):

/usr/lib/uudemon.day	perform once per day
/usr/lib/uudemon.hr	perform once per hour
/usr/lib/uudemon.wk	perform once per week

autoconfig(1M)

Add a cross-reference to newunix(1M) in the SEE ALSO section.

brc(1M)

The system initialization script sysinitrc is part of brc(1).

Add to the SYNOPSIS the line /etc/sysinitrc.

Change the first paragraph of the DESCRIPTION to read:

init executes sysinitrc, brc, bcheckrc, and rc at system initialization, via entries in /etc/inittab. sysinitrc is executed before init starts up its initial level. The others are executed when the system is changed out of single-user mode. powerfail executes whenever a system power failure is detected.

Add a second paragraph to the DESCRIPTION, as follows:

sysinitrc

sysinitrc performs various system initialization tasks, including setting the internal clock, checking the root file system, setting host and domain names, and running autoconfiguration.

Change the paragraph on powerfail to read as follows:

powerfail is invoked when the system detects a power failure. It performs any last-minute activities as desired before powering down.

Add to the FILES section these entries:

`/etc/sysinitrc`

`/etc/sethost`

`/etc/setmactime`

Add to the SEE ALSO section references to `startup`(1M), `autoconfig`(1M), and `query`(1M).

mount(1M)

Note that a space is required between all flag options and their arguments.

Also, the options `quota` and `noquota` should be described under MOUNT FLAG OPTIONS, `-o`, options valid on all file systems:

`quota`	usage limits enforced
`noquota`	usage limits not enforced

In the same section, the default should be `(rw,noquota)`, and not `(rw,suid)` as described.

You should move the description of the `hide` option to the man page for `fstab`(4), as an example of a `mount` option (valid in the `mnt_opts` field) because this option is not particularly useful from the `mount` command, but should be used within entries in the file `/etc/fstab`.

pname(1M)

The man page for pname(1M) describes a `-f` option. This option is currently unsupported. Also, there is a typographical error in the same description. The sentence should read as follows:

If a partition is already associated with the specified slice, pname will first unrecognize that partition.

trpt(1M)

The man page for `trpt`(1M) states that the binary can be found in `/etc/trpt`. The correct pathname is `/usr/etc/trpt`. The SYNOPSIS should also indicate the full pathname of this command.

gethostent(3N)

The man page for gethostent(3N) is incorrect. Remove all references to gethostent, sethostent, and endhostent. File the page under gethostbyaddr(3N).

rexec(3N)

This page incorrectly refers to gethostent(3N). A/UX does not support this call. The first line of the DESCRIPTION should read, "rexec looks up the host referenced by **ahost* using gethostbyname(3N)."

mastermind(6)

There is no man page for /usr/games/mastermind. This program plays the game of Mastermind. When invoked, it will prompt you to decide if you want instructions.

Replacement pages

The following pages are replacements for pages printed in the A/UX reference manuals. Insert them alphabetically by section into the appropriate manuals.

NAME

`checkinstall` – check installation of boards

SYNOPSIS

```
/etc/checkinstall  ethertalk
```

DESCRIPTION

`checkinstall` performs a quick test to see if the named board has been installed or not. The only board type currently supported is the Apple EtherTalk board, which is indicated by the argument `ethertalk`.

FILES

`/etc/checkinstall`

SEE ALSO

`etheraddr`(1M).

NAME

enscript – convert text files to POSTSCRIPT format for printing

SYNOPSIS

enscript [–12BGghKklmoqRr] [–L*lines*] [–f*font*] [–F*hfont*] [–b*header*] [–p*out*] [*spoolopts*] [*files*]

DESCRIPTION

enscript reads plain text files, converts them to POSTSCRIPT format, and spools them for printing on a POSTSCRIPT printer. Fonts, headings, and limited formatting options may be specified.

For example:

```
enscript -paleph boring.txt
```

processes the file called boring.txt for POSTSCRIPT printing, writing the output to the file aleph.

```
enscript -2r boring.c
```

prints a two-up landscape listing of the file called boring.c on the default printer (see below).

Font specifications have two parts: A font name as known to POSTSCRIPT (e.g., Times-Roman, Times-Roman BoldItalic , Helvetica, Courier), and a point size (1 point=1/72 inch). So, Courier-Bold8 is 8 point Courier Bold, Helvetica12 is 12 point Helvetica.

The environment variable ENSCRIPT may be used to specify defaults. The value of ENSCRIPT is parsed as a string of arguments before the arguments that appear on the command line. For example

```
ENSCRIPT='-fTimes-Roman8'
```

sets your default body font to 8 point Times Roman.

The possible options are:

–2	set in two columns.
–1	set in one column (the default).
–r	rotate the output 90 degrees (landscape mode). This is good for output that requires a wide page or for program listings when used in conjunction with –2. "enscript –2r *files*" is a nice way to get program listings.
–R	don't rotate, also known as portrait mode (the default).

`-G` print in gaudy mode: causes page headings, dates, page numbers to be printed in a flashy style, at some slight performance expense.

`-l` simulate a line printer: make pages 66 lines long and omit headers.

`-B` omit page headings.

`-b`*header*
sets the string to be used for page headings to *header*. The default header is constructed from the file name, its last modification date, and a page number.

`-L`*lines* set the maximum number of lines to output on a page. `enscript` usually computes how many to put on a page based on point size, and may put fewer per page than requested by *lines*.

`-f`*font* sets the font to be used for the body of each page. Defaults to Courier10, unless two column rotated mode is used, in which case it defaults to Courier7.

`-F`*hfont* sets the font to be used for page headings. Defaults to `Courier-Bold10`.

`-p`*out* causes the POSTSCRIPT file to be written to the named file rather than being spooled for printing. As a special case, `-p -` will send the POSTSCRIPT to the standard output.

`-g` enables the printing of files containing non-printing characters. Any file with more than a small number of non-printing characters is suspected of being garbage, and is not printed unless this option is used.

`-o` If `enscript` cannot find characters in a font, the missing characters are listed.

`-q` causes `enscript` to be quiet about what it is doing. `enscript` won't report about pages, destination, omitted characters, etc. Fatal errors are still reported to the standard error output.

`-k` enables page prefeed (if the printer supports it). This allows simple documents (e.g., program listings in one font) to print somewhat faster by keeping the printer running between pages.

`-K` disable page prefeed (the default).

-h suppress printing of job burst page.

ENVIRONMENT

ENSCRIPT	string of options to be used by enscript.
PSLIBDIR	path name of a directory to use instead of /usr/lib/ps for enscript prologue and font metric files.
PSTEMPDIR	path name of temporary directory to use instead of XPSTEMDIRX of spooled temporary files.
LPDEST	the name of a printer for lp to use. If LPDEST is not set, enscript will spool to a printer class named PostScript.

FILES

/usr/bin/enscript	
/usr/lib/ps/*.afm	font metrics files.
/usr/lib/ps/enscript.pro	prologue for enscript files.

SEE ALSO

cancel(1), lp(1), lpr(1), lprm(1), lpstat(1), pr(1), ps630(1), getopt(3).

FEATURES

Options and the ENSCRIPT environment string are parsed in getopt(3) fashion.

BUGS

Long lines are truncated. Line truncation may be off by a little bit as printer margins vary. There should be a "wrap" option and multiple (truncated or wrapped) columns.

NAME

`makedev` – prepare troff description files

SYNOPSIS

`makedev` *files*

DESCRIPTION

`makedev` reads description files about a particular device and converts them into a form suitable for reading by `troff`(1). Input to `makedev` is in the format described in `font`(5).

FILES

`/usr/bin/makedev`

SEE ALSO

`troff`(1), `font`(5).
A Typesetter-independent TROFF, Brian W. Kernighan (Bell Laboratories, 1982)
Adventures with Typesetter-Independent TROFF, Mark Kahrs and Lee Moore (University of Rochester TR 159, 1985)

NAME

nslookup – query name servers interactively

SYNOPSIS

nslookup
nslookup – *server*
nslookup *host-to-find* [*server*]

DESCRIPTION

nslookup is a program which queries DARPA Internet domain name servers.

server is a either the host name or address for a name server.

nslookup has two modes: interactive and non-interactive. Interactive mode allows the user to query the name server for information about various hosts and domains or print a list of hosts in the domain. Non-interactive mode is used to print just the name and Internet address of a host or domain.

Interactive mode is entered in the following cases:

a) when no arguments are given (the default name server will be used), and

b) when the first argument is a hyphen (–) and the second argument is the host name of a name server.

Non-interactive mode is used when the name of the host to be looked up is given as the first argument. The optional second argument specifies a *server*.

INTERACTIVE COMMANDS

Commands may be interrupted at any time by typing a CONTROL-c. To exit, enter the end-of-file signal, CONTROL-d. The command line length must be less than 80 characters.

Note: an unrecognized command will be interpreted as a host name.

host [*server*]
Look up information for *host* using the current default server, or using *server* if it is specified.

server *domain*
lserver *domain*
Change the default server to *domain*. lserver uses the

initial server to look up information about *domain* while `server` uses the current default server. If an authoritative answer can't be found, the names of servers that might have the answer are returned.

`root`
: Changes the default server to the server for the root of the domain name space. Currently, the host `sri-nic.arpa` is used. (This command is a synonym for the `lserver sri-nic.arpa`.) The name of the root server can be changed with the `set root` command.

`finger` [*name*] [>*filename*]
`finger` [*name*] [>>*filename*]
: Connects with the finger server on the current host. The current host is defined when a previous lookup for a host was successful and returned address information (see the `set querytype=A` command). *name* is optional. > and >> can be used to redirect output in the usual manner.

`ls` *domain* [>*filename*]
`ls` *domain* [>>*filename*]
`ls -a` *domain* [>*filename*]
`ls -a` *domain* [>>*filename*]
`ls -h` *domain* [>*filename*]
`ls -h` *domain* [>>*filename*]
: List the information available for *domain*. The default output contains host names and their Internet addresses. The `-a` option lists aliases of hosts in the domain. The `-h` option lists CPU and operating system information for the domain. When output is directed to a file, hash marks are printed for every 50 records received from the server.

`view` *filename*
: Sorts and lists the output of the `ls` command with `more`(1).

`help`
`?` Prints a brief summary of commands.

`set` *keyword*[=*value*]

This command is used to change state information that affects the lookups. Valid keywords are:

`all`

Prints the current values of the various options to `set`. Information about the current default server and host is also printed.

`[no]debug`

Turn debugging mode on. A lot more information is printed about the packet sent to the server and the resulting answer.
(Default = `nodebug`, abbreviation = `[no]deb`)

`[no]defname`

Append the default domain name to every lookup.
(Default = `nodefname`, abbreviation = `[no]def`)

`domain=`*name*

Change the default domain name to *name*. The default domain name is appended to all lookup requests if the `defname` option has been set.
(Default = value in `/etc/resolv.conf`, abbreviation = `do`)

`querytype=`*value*

Change the type of information returned from a query to one of:

`A`	the host's Internet address (the default).
`CNAME`	the canonical name for an alias.
`HINFO`	the host CPU and operating system type.
`MD`	the mail destination.
`MX`	the mail exchanger.
`MG`	the mail group member.
`MINFO`	the mailbox or mail list information.
`MR`	the mail rename domain name.

Other types specified in the RFC883 document are valid but aren't very useful.
(Abbreviation = `q`)

`[no]recurse`

Tell the name server to query other servers if it does not have the information.

(Default = `recurse`, abbreviation = `[no]rec`)

`retry=`*number*
: Set the number of retries to *number*. When a reply to a request is not received within a certain amount of time (changed with `set timeout`), the request is resent. The retry value controls how many times a request is resent before giving up.
(Default = 2, abbreviation = `ret`)

`root=`*host*
: Change the name of the root server to *host*. This affects the `root` command.
(Default = `sri-nic.arpa`, abbreviation = `ro`)

`timeout=`*number*
: Change the time-out interval for waiting for a reply to *number* seconds.
(Default = `10` seconds, abbreviation = `t`)

`[no]vc`
: Always use a virtual circuit when sending requests to the server.
(Default = `novc`, abbreviation = `[no]v`)

TUTORIAL

The domain name space is tree-structured and currently has five top-level domains:

- `com` (for commercial establishments)
- `edu` (for educational institutions)
- `gov` (for government agencies)
- `org` (for not for profit orginizations)
- `mil` (for MILNET hosts)

If you are looking for a specific host, you need to know something about the host's organization in order to determine the top-level domain it belongs to. For instance, if you want to find the Internet address of a machine at UCLA, do the following:

a) Connect with the root server using the `root` command. The root server of the name space has knowledge of the top-level domains.

b) Since UCLA is a university, its domain name is `ucla.edu`. Connect with a server for the `ucla.edu` domain with the command `server ucla.edu`. The response will print the names of hosts that act as servers for the domain `ucla.edu`. Note that the root server does not have information about `ucla.edu` but knows the names and addresses of hosts that do. All future queries will be sent to the UCLA name server.

c) To request information about a particular host in the domain, type the host name. To request a listing of hosts in the UCLA domain, use the `ls` command. The `ls` command requires a domain name (in this case, `ucla.edu`) as an argument.

Note that if you are connected with a name server that handles more than one domain, all lookups for host name must be fully specified with its domain. For instance, the domain `harvard.edu` is served by `seismo.css.gov`, which also services the `css.gov` and `cornell.edu` domains. A lookup request for the host `aiken` in the `harvard.edu` domain must be specified as `aiken.harvard.edu`. However, the `set domain=`*name* and `set defname` commands can be used to automatically append a domain name to each request.

After a successful lookup of a host, use the `finger` command to see who is on the system or to finger a specific person. To get other information about the host, use the `set querytype=`*value* command to change the type of information desired and request another lookup. (`finger` requires *value* to be `A`.)

DIAGNOSTICS

If the lookup request was not successful, an error message is printed. Possible errors are:

`Time-out`
The server did not respond to a request after a certain amount of time (changed with `set timeout=`*value*) and a certain number of retries (changed with `set retry=`*value*).

`No information`
Depending on the query type set with the `set querytype` command, no information about the host was available, though the host name is valid.

`Non-existent domain`
The host or domain name does not exist.

`Connection refused`
`Network is unreachable`
The connection to the name or finger server could not be made at the current time. This error commonly occurs with

`finger` requests.

`Server failure`
The name server found an internal inconsistency in its database and could not return a valid answer.

`Refused`
The name server refused to service the request.

`Format error`
The name server found that the request packet was not in the proper format. This error should not occur. It would indicate a bug in the program.

FILES

`/etc/bind/tools/nslookup`	
`/etc/resolv.conf`	initial domain name and name server addresses

SEE ALSO

`named`(1M), `resolver`(4).
RFC-882, RFC-883 (DNN Network Information Center, SRI International)

AUTHOR

Andrew Cherenson

NAME

query – query the user for input

SYNOPSIS

`query [ -t [seconds] ] [ -r [response] ] [ -m ]`

DESCRIPTION

By default, `query` reads a line from standard input and echoes it to standard output. Options include:

`–t [`*seconds*`]`
: Timeout after *seconds* seconds. If no input has been seen by this time, `query` will echo the default response value to standard output and standard error.

`–r [`*response*`]`
: Change the default response to *response*. The default response is `y` if not set with this option. The `–r` option is only useful in conjunction with `–t`.

`–m` Watch for a mouse click. If the mouse does get clicked, exit status 2 is returned. Note: This option will be ignored if any other program (such as a toolbox application) is currently using the mouse.

DIAGNOSTICS

Exit status is 0 if everything is OK, 1 for usage error, 2 if mouse is pressed when `query –m` is in use.

FILES

`/etc/query`

SEE ALSO

`line(1)`.

NAME

`rcnvt` – resource file format converter

SYNOPSIS

`rcnvt[ -s] [-f ] -i`*input-file* `-o`*output-file*

DESCRIPTION

`rcnvt` converts Macintosh resource files from the old `.res` format to the new AppleDouble or AppleSingle formats. The command line options and their meanings are

`-i` *input-file*
: Specify the name of the old format file to be converted. The `.res` extension should not be included.

`-o` *output-file*
: Specify the name of the output file, which will be in AppleSingle or AppleDouble format. The `%` prefix on AppleDouble resource forks will be automatically appended by the program and should not be included here.

`-s` Create an AppleSingle format output file instead of the default Apple Double format file. AppleSingle combines both the resource and data forks into a Single A/UX file, and is most useful when the Resource fork seldom or never changes, or on files with no data fork. Use of AppleSingle format is very inefficient when both the resource and data forks are frequently expanded.

`-f` Suppress errors when creating AppleSingle format files that have no data fork.

AppleSingle is particularly nice for executable Macintosh object files. Directory listings look much cleaner because each Macintosh file maps to a single A/UX file with no `%` prefix or `.res` suffix.

`rcnvt` can also be used when transferring a resource file from the native Macintosh Environment to A/UX. If this transfer is done via the `mfs`(1) command, AppleSingle or AppleDouble format files are created. However, if the transfer is done via a terminal emulator program, the file created will probably just contain a copy of the Macintosh file's resource fork. In this case, `rcnvt` can be used to convert the file to either AppleSingle or AppleDouble format. Note that this technique of file transfer will not preserve the Macintosh file's type or creator. These can be set with the `settc`(1) command.

FILES

/usr/toolboxbin/rcnvt

SEE ALSO

mfs(1), settc(1).

NAME

remlogin – remote sign on

SYNOPSIS

remlogin

DESCRIPTION

The remlogin command is used when a user initially signs on from a remote host.

When remlogin is invoked from a virtual terminal server process, it asks for a user name, and, if appropriate, a password. Echoing is turned off (if possible) during the typing of the password, so it will not appear on the written record of the session.

After a successful login, accounting files are updated and the user is informed of the existence of mail. The message of the day is printed, as is the time of his last login. Both are suppressed if he has a .hushlogin file in his home directory; this is mostly used to make life easier for non-human users, such as uucp.

remlogin initializes the user and group IDs and the working directory, then executes a command interpreter (usually csh(1)) according to specifications found in a password file. Argument 0 of the command interpreter is the name of the command interpreter with a leading dash (–).

remlogin also modifies the environment environ(7) with information specifying home directory, command interpreter, terminal type (if available) and user name.

If the file /etc/nologin exists, login prints its contents on the user's psuedo terminal and exits. This is used by shutdown(1M) to stop users logging in when the system is about to go down.

FILES

/usr/spool/mail/*	
/etc/utmp	accounting
/usr/adm/wtmp	accounting
/usr/spool/mail/*	mail
/etc/motd	message-of-the-day
/etc/passwd	password file
/etc/nologin	stops logins
.hushlogin	makes login quieter

SEE ALSO

mail(1), passwd(1), getty(1M), rlogind(1M), telnetd(1M), init(1M), shutdown(1M), rlogin(1N), passwd(5), environ(7).

DIAGNOSTICS

`Login incorrect`
: if the name or the password is bad.

`No Shell`
: cannot open password file

`no directory`
: consult a system administrator.

BUGS

`remlogin` uses two undocumented options. `-r` is used by the remote login server, `rlogind`(1M) to force `remlogin` to enter into an initial connection protocol. `-h` is used by `telnetd`(1M) and other servers to list the host from which the connection was received.

NAME

`screenrestore` – restore the A/UX Initial Console Emulator screen

SYNOPSIS

`screenrestore`

DESCRIPTION

`screenrestore` restores the A/UX Initial Console Emulator screen bitmap and resets the mac2 vt100-like console emulator to an initialized state. `screenrestore` is used to clean up after certain graphics programs (e.g. X Windows) which may leave the screen in a strange state.

FILES

`/usr/bin/screenrestore`

NAME

term – emulate a vt100 in A/UX Toolbox windows

SYNOPSIS

term

DESCRIPTION

term is a terminal emulator program, built on top of the A/UX Toolbox. It emulates a DEC vt100. term uses the standard Macintosh user interface and supports multiple, resizable windows. The source code and makefile can be found in /usr/lib/mac/examples/term. This is a good example of how to write software that uses the A/UX Toolbox.

FILES

/usr/toolboxbin/term	
/usr/lib/mac/examples/term/*	source code

SEE ALSO

toolboxdaemon(1M).
A/UX Toolbox Guide

NAME

ethers, ether_ntoa, ether_aton, ether_ntohost, ether_hostton, ether_line – Ethernet address mapping operations

SYNOPSIS

```
#include <sys/types.h>
#include <sys/socket.h>
#include <net/if.h>
#include <netinet/in.h>
#include <netinet/if_ether.h>

char *
ether_ntoa(e)
      struct ether_addr *e;

struct ether_addr *
ether_aton(s)
      char *s;

ether_ntohost(hostname, e)
      char *hostname;
      struct ether_addr *e;

ether_hostton(hostname, e)
      char *hostname;
      struct ether_addr *e;

ether_line(l, e, hostname)
      char *l;
      struct ether_addr *e;
      char *hostname;
```

DESCRIPTION

These routines are useful for mapping 48-bit Ethernet numbers to their ASCII representations or their corresponding host names, and vice versa.

The function `ether_ntoa` converts a 48-bit Ethernet number pointed to by `e` to its standard ACSII representation; it returns a pointer to the ASCII string. The representation is of the form: *x:x:x:x:x:x*: where *x* is a hexadecimal number between 0 and 255. The function `ether_aton` converts an ASCII string in the standard representation back to a 48-bit Ethernet number; the function returns NULL if the string cannot be scanned successfully.

The function `ether_ntohost` maps an Ethernet number (pointed to by `e`) to its associated hostname. The string pointed to by *hostname* must be long enough to hold the hostname and a null character. The function returns zero upon success and non-zero upon failure. Inversely, the function `ether_hostton` maps a hostname string to its corresponding Ethernet number; the function modifies the Ethernet number pointed to by `e`. The function also returns zero upon success and non-zero upon failure.

The function `ether_line` scans a line (pointed to by `l`) and sets the hostname and the Ethernet number (pointed to by `e`). The string pointed to by *hostname* must be long enough to hold the hostname and a null character. The function returns zero upon success and non-zero upon failure. The format of the scanned line is described by `ethers`(4).

FILES

`/etc/ethers`	
`/etc/ethers.byaddr`	Yellow Pages control file
`/etc/ethers.byname`	Yellow Pages control file

SEE ALSO

`ethers`(4).

NAME

HOSTNAME – hostname and domainname database

DESCRIPTION

HOSTNAME resides in the /etc directory and consists of one line containing the following items of information

hostname domainname

Items are separated by any number of blanks and/or tabs. There must be no white space at the beginning of the line.

hostname is the name of the local host machine and *domainname* is the name of the Yellow Pages domain on which the local host resides.

EXAMPLE

```
magic       apple
```

FILES

/etc/HOSTNAME

SEE ALSO

hostname(1), domainname(1), chgnod(1M).
A/UX Installation Guide
RFC-882, RFC-883, RFC-920, RFC-921, RFC-952, RFC-953, RFC-973, RFC-974 (DNN Network Information Center, SRI International)

NAME

NETADDRS – network address database

DESCRIPTION

The NETADDRS file resides in /etc and contains information regarding the network addresses of each EtherTalk board on the local machine. For each board, a single line should be present with the following items of information:

unit-number internet-address broadcast-address netmask

Items are separated by any number of blanks and/or tab characters. Lines must not begin with blanks or tabs. *netmask* should be blank if subnets are not being supported.

EXAMPLE

The following is a sample NETADDRS file for a machine on two networks; only the second is subnetted.

```
0    89.53        89.0
1    91.1.0.48    91.1.0.0    255.255.0.0
```

FILES

/etc/NETADDRS

SEE ALSO

autoconfig(1M), ifconfig(1M).
A/UX Network System Administration
RFC-917, RFC-922, RFC-944, RFC-950 (DDN Network Information Center , SRI International)

NAME

ethers – Ethernet address to hostname database or YP domain

DESCRIPTION

The /etc/ethers file contains information regarding the known (48 bit) Ethernet addresses of hosts on the Internet. For each host on an Ethernet, a single line should be present with the following items of information:

ethernet-address hostname

Items are separated by any number of blanks and/or tabs. Use # to introduce a single line or midline comment.

The standard form for *ethernet-address* is *x:x:x:x:x:x:* where *x* is a hexadecimal number between 0 and 255, representing one byte. The address bytes are always in network order. *hostname* may contain any printable character other than a space, tab, newline, or comment character. The hostnames in the ethers file should correspond to the hostnames in the /etc/hosts file (see hosts(4)).

The *ether_line*() routine from the Ethernet address manipulation library, ethers(3N) may be used to scan lines of the ethers file.

FILES

/etc/ethers

SEE ALSO

ethers(3N), hosts(4).

NAME

rpc – rpc program number data base

SYNOPSIS

/etc/rpc

DESCRIPTION

The The rpc file contains user readable names that can be used in place of rpc program numbers. Each line has the following items of information:

server-name program-number [*alias...*]

Items are separated by any number of blanks and/or tab characters. Use # to indicate the beginning of a comment; characters up to the end of the line are not interpreted by routines which search the file.

EXAMPLE

```
#
#     rpc    1.1    86/07/07
#
portmapper       100000  portmap sunrpc
rstatd           100001  rstat rup perfmeter
rusersd          100002  rusers
nfs              100003  nfsprog
ypserv           100004  ypprog
mountd           100005  mount showmount
ypbind           100007
walld            100008  rwall shutdown
yppasswdd        100009  yppasswd
etherstatd       100010  etherstat
rquotad          100011  rquotaprog quota rquota
sprayd           100012  spray
3270_mapper      100013
rje_mapper       100014
selection_svc    100015  selnsvc
database_svc     100016
rexd             100017  rex
alis             100018
sched            100019
llockmgr         100020
nlockmgr         100021
x25.inr          100022
statmon          100023
status           100024
```

FILES

`/etc/rpc`

SEE ALSO

rpc(3N).

NAME

icmp – Internet Control Message Protocol

SYNOPSIS

None; included automatically with inet(5F).

DESCRIPTION

The Internet Control Message Protocol, ICMP, is used by gateways and destination hosts which process datagrams to communicate errors in datagram-processing to source hosts. The datagram level of Internet is discussed in ip(5P). ICMP uses the basic support of IP as if it were a higher level protocol; however, ICMP is actually an integral part of IP. ICMP messages are sent in several situations; for example: when a datagram cannot reach its destination, when the gateway does not have the buffering capacity to forward a datagram, and when the gateway can direct the host to send traffic on a shorter route.

The Internet protocol is not designed to be absolutely reliable. The purpose of these control messages is to provide feedback about problems in the communication environment, not to make IP reliable. There are still no guarantees that a datagram will be delivered or that a control message will be returned. Some datagrams may still be undelivered without any report of their loss. The higher level protocols which use IP must implement their own reliability mechanisms if reliable communication is required.

The ICMP messages typically report errors in the processing of datagrams; for fragmented datagrams, ICMP messages are sent only about errors in handling fragment 0 of the datagram. To avoid the infinite regress of messages about messages etc., no ICMP messages are sent about ICMP messages. ICMP may however be sent in response to ICMP messages (for example, ECHOREPLY). There are eleven types of ICMP packets which can be received by the system. They are defined in this excerpt from <netinet/ip_icmp.h>, which also defines the values of some additional codes specifying the cause of certain errors. (Comments have been stripped for this listing.)

```
/*
 * Definition of type and code field values
 */
#define ICMP_ECHOREPLY        0
#define ICMP_UNREACH          3
#define  ICMP_UNREACH_NET     0
```

```
#define   ICMP_UNREACH_HOST      1
#define   ICMP_UNREACH_PROTOCOL 2
#define   ICMP_UNREACH_PORT      3
#define   ICMP_UNREACH_NEEDFRAG 4
#define   ICMP_UNREACH_SRCFAIL  5
#define  ICMP_SOURCEQUENCH       4
#define  ICMP_REDIRECT           5
#define   ICMP_REDIRECT_NET      0
#define   ICMP_REDIRECT_HOST     1
#define   ICMP_REDIRECT_TOSNET   2
#define   ICMP_REDIRECT_TOSHOST 3
#define  ICMP_ECHO               8
#define  ICMP_TIMXCEED           11
#define   ICMP_TIMXCEED_INTRANS 0
#define   ICMP_TIMXCEED_REASS   1
#define  ICMP_PARAMPROB          12
#define  ICMP_TSTAMP             13
#define  ICMP_TSTAMPREPLY        14
#define  ICMP_IREQ               15
#define  ICMP_IREQREPLY          16
```

Arriving ECHO and TSTAMP packets cause the system to generate ECHOREPLY and TSTAMPREPLY packets. IREQ packets are not yet processed by the system, and are discarded. UNREACH, SOURCEQUENCH, TIMXCEED and PARAMPROB packets are processed internally by the protocols implemented in the system, or reflected to the user if a raw socket is being used; see ip(5P). REDIRECT, ECHOREPLY, TSTAMPREPLY and IREQREPLY are also reflected to users of raw sockets. In addition, REDIRECT messages cause the kernel routing tables to be updated; see routing(5N).

SEE ALSO

inet(5F), ip(5P).
Internet Control Message .'rotocol, RFC792, J. Postel, USC-ISI

BUGS

IREQ messages are not pro essed properly: the address fields are not set.

Messages which are source ro ıted are not sent back using inverted source routes, but rather go back through the normal routing mechanisms.

NAME

arp – address resolution display and control

SYNOPSIS

/etc/arp *hostname*
/etc/arp -a [*unix*] [*kmem*]
/etc/arp -d *hostname*
/etc/arp -s *hostname ether-addr* [temp] [pub]
/etc/arp -f *filename*

DESCRIPTION

The arp program displays and modifies the Internet-to-Ethernet address translation tables used by the address resolution protocol (arp(5)).

With no flags, the program displays the current ARP entry for *hostname*. The host may be specified by name or by number, using Internet dot notation. With the -a flag, the program displays all of the current ARP entries by reading the table from the file kmem (default /dev/kmem) based on the kernel file unix (default /unix).

With the -d flag, a superuser may delete an entry for the host called *hostname*.

The -s flag is given to create an ARP entry for the host called *hostname* with the Ethernet address *ether-addr*. The Ethernet address is given as six hex bytes separated by colons. The entry will be permanent unless the word temp is given in the command. If the word pub is given, the entry will be "published"; i.e., this system will act as an ARP server, responding to requests for *hostname* even though the host address is not its own.

The -f flag causes the file *filename* to be read and multiple entries to be set in the ARP tables. Entries in the file should be of the form

hostname ether-addr [temp] [pub]

FILES

/etc/arp
/dev/kmem

SEE ALSO

inet(3N), arp(5), ifconfig(1M).

NAME

etheraddr – get an Ethernet address

SYNOPSIS

/etc/etheraddr [*slot*]

DESCRIPTION

etheraddr prints the Ethernet address stored in ROM on the board in slot number *slot*.

DIAGNOSTICS

etheraddr exits with the return status 0 if an Ethernet interface and valid ROM are available. A nonzero exit status indicates failure to find or read an Ethernet address for the host.

FILES

/etc/etheraddr

SEE ALSO

slots(3X), ae(5), arp(5P), inet(5F), intro(5).

NAME

eu – update autorecovery files

SYNOPSIS

/etc/eu *file*

DESCRIPTION

eu is used to maintain the files needed by autorecovery(8). It copies *file* to the eschatology partition(s) and updates the relevant entry in /etc/eschatology/init2files. If *file* is not found in /etc/eschatology/init2files, an entry as described in cml(4) will be created.

To prevent inconsistent updates while eu is running, a lockfile, /etc/eschatology/FCML.lock, is used to single-thread the updates to the file systems and the cml(4) file. Once eu is complete, this lockfile is removed.

FILES

/etc/eu	
/etc/eschatology/init2files	the data base
/etc/eschatology/FCML.lock	the lock file

SEE ALSO

autorecovery(8), escher(1M), cml(4).

NAME

eupdate – update important files for autorecovery

SYNOPSIS

```
/etc/eupdate
```

DESCRIPTION

eupdate updates appropriate system files for autorecovery use. This command should be used after a machine has been reconfigured with autoconfig(1M) or after modification of important relevant files (see below).

FILES

```
/etc/HOSTNAME
/etc/NETADDRS
/etc/eupdate
/etc/inittab
/etc/startup.d/BNET
/etc/startup.d/ae6
/unix
```

SEE ALSO

autoconfig(1M), eu(1M), autorecovery(8).

NAME

`fingerd` – remote user information server

SYNOPSIS

`/usr/etc/in.fingerd`

DESCRIPTION

`fingerd` is a simple protocol based on RFC742 that provides an interface to the `name` and `finger` programs at several network sites. The program reports status information about either the system at the moment or a particular person in depth. There is no required format and the protocol consists mostly of specifying a single ''command line''.

`fingerd` listens for TCP requests at port 79. Once connected it reads a single command line terminated by a <CR><LF> which is passed to `finger(1)`. `fingerd` closes its connections as soon as the output is finished.

If the line is null (i.e. just a <CR><LF> is sent) then `finger` returns a ''default'' report that lists all people logged into the system at that moment.

If a login name is specified (so `fingerd` receives eric<CR><LF>), then more extensive information is provided for that user, whether logged in or not. Allowable user names in the command line include both login names and user names. If a name is ambiguous, all possible derivations are returned.

FILES

`/usr/etc/in.fingerd`

SEE ALSO

`finger(1)`.
RFC742 (DNN Network Information Center, SRI International)

BUGS

Connecting directly to the server from a TIP or an equally narrow-minded TELNET-protocol user program can result in meaningless attempts at option negotiation being sent to the server, which will foul up the command line interpretation. `fingerd` should be enhanced to filter out IAC's and perhaps even respond negatively (IAC WON'T) to all option commands received.

NAME

newunix – prepare for new kernel configuration

SYNOPSIS

/etc/newunix [bnet] [nfs] [nonet] [toolbox] [notoolbox]

DESCRIPTION

newunix begins the process of configuring a new kernel by installing (or uninstalling) the appropriate scripts and driver object files needed by autoconfig(1M). The appropriate argument to newunix depends on the type of kernel desired: basic networking (bnet), Network File System (nfs), A/UX toolbox (toolbox), non-networking (nonet), no toolbox capabilities (notoolbox).

In order to complete the kernel configuration process, autoconfig(1M) should be run after newunix.

FILES

/etc/newunix	
/etc/boot.d/*	driver object files
/etc/install.d/*	installation scripts
/etc/master.d/*	script files
/etc/startup.d/*	startup programs
/etc/uninstall.d/*	uninstallation scripts

SEE ALSO

autoconfig(1M).

NAME

ping – network debugging

SYNOPSIS

/usr/etc/ping *host* [*timeout*]

DESCRIPTION

ping repeatedly sends an icmp echo packet to *host* and reports whether or not a reply was received. It keeps trying until *timeout* seconds have elapsed, or an answer is received. The default timeout is 20 seconds. The *host* argument can be a name or an internet address.

FILES

/usr/etc/ping

SEE ALSO

icmp(5P).

NAME

revnetgroup – reverse the netgroup file

SYNOPSIS

/etc/yp/revnetgroup [–u] [–h]

DESCRIPTION

revnetgroup reverses the netgroup file. Options are

–u reverse by username

–h reverse by hostname

Each line in the output file will begin with a key formed by concatenating the host or user name with the domain name. The key will be followed by a tab, then the comma-separated, newline-terminated list of groups to which the user or host belongs.

Exception: Groups to which everyone belongs (universal groups) will not be included in the list. The universal groups will be listed under the special name *.

NOTE

revnetgroup is a filter used in updating the /etc/yp data bases. It is not expected to be of general utility.

FILES

/etc/yp/revnetgroup
/etc/netgroup

SEE ALSO

ypmake(1M), netgroup(4).

NAME

startup – run startup programs at boot time

SYNOPSIS

/etc/startup

DESCRIPTION

startup is a shell script, called from /etc/sysinitrc, which runs a set of startup routines to initialize autoconfigured modules that are part of the kernel. An example startup script is /etc/startup.d/BNET which initializes the loop interface lo0.

NOTE

startup is called from /etc/sysinitrc at system startup. It is not expected to be of general utility.

FILES

/etc/startup
/etc/startup.d/*

SEE ALSO

autoconfig(1M), sysinitrc(1M).

NAME

`stdhosts` – convert Internet addresses to standard form

SYNOPSIS

`/etc/yp/stdhosts` *file*

DESCRIPTION

`stdhosts` converts Internet addresses to a standard form. Addresses are read from a file (usually `/etc/hosts`).

NOTE

`stdhosts` is a filter used in updating the `/etc/yp` data bases. It is not expected to be of general utility.

FILES

`/etc/yp/stdhosts`	
`/etc/hosts`	the host table

SEE ALSO

`ypmake`(1M), `hosts`(4).

NAME

toolboxdaemon – set up for and clean up after A/UX Toolbox programs

SYNOPSIS

toolboxdaemon

DESCRIPTION

toolboxdaemon is a daemon that should be running in the background whenever you plan to run programs that use the A/UX Toolbox.

When an A/UX Toolbox program is started, toolboxdaemon sets up the shared data and shared text segments. After an A/UX Toolbox program exits, toolboxdaemon cleans up the structures allocated to the program.

To have toolboxdaemon started every time the machine enters multiuser mode, change the tb0 entry in the file /etc/inittab. The default system inittab file has this entry:

```
tb0:2:off:/etc/toolboxdaemon > /dev/syscon 2>&1
```

Change the word "off" to "respawn," so that the line reads

```
tb0:2:respawn:/etc/toolboxdaemon > /dev/syscon 2>&1
```

If this line does not appear in your /etc/inittab file, add it.

When you are in the A/UX shell, you can determine whether toolboxdaemon is running with the ps(1) command:

```
ps -ef
```

EXAMPLE

The command

```
/etc/toolboxdaemon &
```

starts toolboxdaemon running in the background.

FILES

/usr/toolboxbin/toolboxdaemon

NAME

uucico, uushell – transfer files queued by uucp or uux

SYNOPSIS

/usr/lib/uucp/uucico [–d*spooldir*] [–g*grade*] [–r*role*] [–R] [–s*system*] [–x*debug*] [–L] [–t*turnaround*]

/usr/lib/uucp/uushell

DESCRIPTION

uucico performs the actual work involved in transferring files between systems. uucp(1C) and uux(1C) merely queue requests for data transfer which uucico processes.

uushell serves as the login shell for the user uucp. uushell sets the environment variable TZ and calls uucico.

The following options are available to uucico.

–d*spooldir*
: Use *spooldir* as the spool directory. The default is /usr/spool/uucp.

–g*grade* Only send jobs of grade *grade* or higher this transfer. The grade of a job is specified when the job is queued by uucp or uux.

–r*role* *role* is either 1 or 0; it indicates whether uucico is to start up in master or slave role, respectively. 1 is used when running uucico by hand or from cron(1M). 0 is used when another system calls the local system. Slave *role* is the default.

–R Reverse roles. When used with the –r1 option, this tells the remote system to begin sending its jobs first, instead of waiting for the local machine to finish.

–s*system*
: Call only system *system*. If –s is not specified, and –r1 is specified, uucico will attempt to call all systems for which there is work. If –s is specified, a call will be made even if there is no work for that system. This is useful for polling.

–x*debug* Turn on debugging at level *debug*. Level 5 is a good start when trying to find out why a call failed. Level 9 is very detailed. Level 99 is absurdly verbose. If *role* is 1 (master), output is normally written to the standard error output. If the standard error output is unavailable, output is written to `/usr/spool/uucp/AUDIT/`*system.* When *role* is `0` (slave), debugging output is always written to the AUDIT file.

`-L` Only call "local" sites. A site is considered local if the device-type field in `L.sys` is one of `LOCAL`, `DIR`or `TCP`.

`-t`*turnaround*
Use *turnaround* as the line turnaround time (in minutes) instead of the default 30. If *turnaround* is missing or `0`, line turnaround will be disabled. After `uucico` has been running in slave role for *turnaround* minutes, it will attempt to run in master role by negotiating with the remote machine. In earlier versions of `uucico`, a transfer of many large files in one direction would hold up mail going in the other direction. With the turnaround code working, the message flow will be more bidirectional in the short term. This option only works with newer `uucicos` and is ignored by older ones.

If `uucico` receives a `SIGFPE` (see `kill(1)`), it will toggle the debugging on or off.

`uucico` is commonly used either of two ways: as a daemon run periodically by `cron(1M)` to call out to remote systems, and as a "shell" for remote systems who call in. For calling out periodically, a typical line in a crontab file would be:

```
0 * * * * /usr/lib/uucp/uucico -r1
```

This will run `uucico` every hour in master role. For each system that has transfer requests queued, `uucico` calls the system, logs in, and executes the transfers. The file `L.sys` is consulted for information about how to log in, while `L-devices` specifies available lines and modems for calling.

For remote systems to dial in, an entry in the `passwd(4)` file must be created, with a login "shell" of `uushell`. For example:

```
nuucp:Password:5:5::/usr/spool/uucppublic:/usr/lib/uucp/uushell
```

The UID for UUCP remote logins is not critical, so long as it differs from the UUCP Administrative login. The latter owns the UUCP files, and assigning this UID to a remote login would be an extreme security hazard.

FILES

/usr/spool/uucp/D.hostnameX/

/usr/lib/uucp/	UUCP internal files/utilities
/usr/lib/uucp/L-devices	Local device descriptions
/usr/lib/uucp/L-dialcodes	Phone numbers and prefixes
/usr/lib/uucp/L.cmds	Remote command permissions list
/usr/lib/uucp/L.sys	Host connection specifications
/usr/lib/uucp/USERFILE	Remote directory tree permissions list
/usr/spool/uucp/	Spool directory
/usr/spool/uucp/AUDIT/*	Debugging audit trails
/usr/spool/uucp/C./	Control files directory
/usr/spool/uucp/D./	Incoming data file directory
/usr/spool/uucp/D.hostname/	Outgoing data file directory
/usr/spool/uucp/D.hostnameX/	Outgoing execution file directory
/usr/spool/uucp/CORRUPT/	Place for corrupted C. and D. files
/usr/spool/uucp/ERRLOG	UUCP internal error log
/usr/spool/uucp/LOGFILE	UUCP system activity log
/usr/spool/uucp/LCK/LCK..*	Device lock files
/usr/spool/uucp/SYSLOG	File transfer statistics log
/usr/spool/uucp/STST/*	System status files
/usr/spool/uucp/TM./	File transfer temp directory

/usr/spool/uucp/X./	Incoming execution file directory
/usr/spool/uucppublic	Public access directory

SEE ALSO

uucp(1C), uuq(1C), uux(1C), uuclean(1M), uuxqt(1M).
D. A. Nowitz and M. E. Lesk, *A Dial-Up Network of UNIX Systems*.
D. A. Nowitz, *Uucp Implementation Description*.

NAME

uuxqt – UUCP execution file interpreter

SYNOPSIS

/usr/lib/uucp/uuxqt [–x*debug*]

DESCRIPTION

uuxqt interprets "execution files" created on a remote system via uux(1C) and transferred to the local system via uucico(1M). When a user uses uux to request remote command execution, it is uuxqt that actually executes the command. Normally, uuxqt is forked from uucico to process queued execution files; for debugging, it may also be run manually by the UUCP administrator.

uuxqt runs in its own subdirectory, /usr/spool/uucp/.XQTDIR. It copies intermediate files to this directory when necessary.

FILES

/usr/spool/uucp/LCK.XQT	
/usr/lib/uucp/L.cmds	Remote command permissions list
/usr/lib/uucp/USERFILE	Remote directory tree permissions list
/usr/spool/uucp/LOGFILE	UUCP system activity log
/usr/spool/uucp/LCK.XQT	uuxqt lock file
/usr/spool/uucp/X./	Incoming execution file directory
/usr/spool/uucp/.XQTDIR	uuxqt running directory

SEE ALSO

uucp(1C), uux(1C), uucico(1M).

NAME

autorecovery – file system repair procedure

DESCRIPTION

Autorecovery facilities check and repair A/UX filesystems. They include a variety of programs, some of which run under the stand-alone shell, and some of which run under A/UX. The preparatory work is done on an incremental basis using the A/UX programs. The actual checking and correction tasks are performed within the stand-alone portion of the boot sequence, and are invoked manually as desired. The checking and correction portions of autorecovery can also run automatically upon booting, if the stand-alone environment has been set up properly.

autorecovery is the name of a stand-alone shell variable that contains the autorecovery command string. This command string is executed whenever you boot A/UX (see sash(8)). The command string has been initially set to echo no autorecovery as an indication that autorecovery is not being run.

To request autorecovery manually, run esch directly from the stand-alone shell. An alternative is to set autorecovery so that the desired form of the esch command is run automatically upon booting.

The A/UX progams that help support the proper functioning of the esch autorecovery function are those in the SEE ALSO section with the designation "1M" (indicating a system administrator command).

SEE ALSO

escher(1M), eu(1M), eupdate(1M), esch(8). sash(8).

NAME

esch – standalone file system repair

SYNOPSIS

esch [-b] [-c *cluster-number*] [-f] [-v]

DESCRIPTION

esch attempts to insure a minimal A/UX file system exists for a multiuser boot. When possible, it will correct bad blocks, repair file system inconsistencies and replace corrupt or missing files. esch is intended to be run when there is reason to suspect that the A/UX file systems have been damaged.

Flag options to esch are:

-b bypasses bad block checking functions.

-c *cluster-number*
allows the user to specify the autorecovery cluster number.

-f does not perform fsck (see fsck(1M)).

-v reports any corrective measures taken.

esch must be run in the standalone environment before the A/UX system is booted. When the system is powered on or A/UX is rebooted, and the boot dialog is cancelled, the standalone shell (see sash(8)) prompt will appear and the esch command line may be entered.

esch may be run automatically by setting the sash(8) variable autorecovery to the desired esch command line.

A hard disk may be divided into partitions. Each partition contains one file system (see fs(4)). Information on all the partitions on a disk is kept in the disk partition map (see dpme(4)) for that disk. One of the fields in a dpme is the cluster number. This is used to identify the group of partitions esch should use.

esch requires that all partitions reside on one disk. esch will read the cluster number from nvram (see nvram(7)) and locate all the partitions in that cluster. A cluster must contain a root partition, a swap partition and at least one autorecovery partition. A cluster may also contain a partition known to esch as the usr partition. There may be multiple autorecovery partitions in a cluster.

Autorecovery file systems contain copies of files that are necessary for a minimal multiuser A/UX system. If new versions of commands, programs or files are installed on the system, the autorecovery file systems should be updated using eu(1M) or escher(1M).

esch will check each file system for bad blocks. This is done by verifying that each block can be read. If possible, any blocks that cannot be read will be spared by the hardware or alt blocked (see altblk(4)) by the software. If neither of these is successful, the block number is added to a list of blocks that is passed to fsck. fsck will add these blocks to a file associated with inode one; this has the effect of removing the blocks from the file system. Checking for bad blocks is a time consuming process. This phase of autorecovery may be omitted using the -b flag option. It is advisable to occasionally run esch without the -b flag option to be sure any bad blocks have been dealt with in the appropriate manner.

esch will then do an fsck on each file system. This fsck is similar to running fsck with the -y flag option. All questionable files will be removed from the file system. If fsck should fail or if the superblock is unreadable, a mkfs (see mkfs(1M)) will be performed on the file system.

After the file systems have been checked for consistency, esch will enter the file check merge (fcm) phase. The configuration master list (see cml(4)) is a list of files required for a multiuser A/UX system. This file gives rules about the attributes of each required file. The file check merge phase of esch will check each file in the cml for conformity to the specified file attributes (size, version, check sum, permissions, etc). If a file does not conform to these rules, esch will attempt to replace it with a copy from an autorecovery file system. If the file in question is found on an autorecovery file system, it must conform to the cml rules or it will not be placed on the root or usr file system.

FILES

/etc/eschatology/init2files

SEE ALSO

escher(1M), eu(1M), eupdate(1M), fsck(1M), mkfs(1M), altblk(4), cml(4), dpme(4), fs(4), autorecovery(8), sash(8).
"System Startup and Shutdown" in *A/UX Local System Administration.*

WARNINGS

esch must *never* be interrupted! Do not power off the system nor push the reset button while esch is running. If esch is interrupted, major file system damage may result or entire file systems may be destroyed.

esch will only attempt to replace files on two file systems. These are the root file system and a file system that is intended to be mounted on /usr. Any other file systems will be ignored by esch.

The superblock of a file system contains information describing the file system. If esch is unable to read the superblock of a file system, or if the superblock has a bad magic number, the file system is not usable and an mkfs will be performed upon the file system. Everything on the file system will be removed! All user files will be gone and cannot be restored since the only files esch restores will be those listed in the cml.

If a file system should become full while esch is copying a replacement file to it, esch will attempt to free up space by deleting files from /lost+found, /tmp, /usr/lost+found or /usr/tmp (depending upon whether the file system is root and/or usr). Subdirectories and their contents will also be removed from these directories. If the directories have been emptied and there is still no room to copy required files, esch will terminate with an error.

Notes

Notes